NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

SIGNAL SYNTHESIS WITH DYNAMICALLY-CHANGED POWER SPECTRAL DENSITY IN A SOFTWARE DEFINED RADIO TRANSMITTER

by

Nikolaos Apostolou

September 2003

Thesis Advisor:	Jovan Lebaric
Second Reader:	David Jenn

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** September 2003	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE: Synthesis of Signal With a Dynamically Changed Power Spectral Density in a Software Defined Radio (SDR) Transmitter			**5. FUNDING NUMBERS**
6. AUTHOR(S) Nikolaos Apostolou			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			**12b. DISTRIBUTION CODE**

13. ABSTRACT ***(maximum 200 words)***

The objective of this thesis is to synthesize signals with a dynamically change power spectral density, in a SDR transmitter, utilizing the most appropriate channels, modulation schemes and transmission rates for communication, based on the noise profile (AWGN plus interferences) of the link, in order to achieve performance within some predefined acceptable levels. The objective is obtained by simulation.

14. SUBJECT TERMS Software Defined Radio, RF Section, IF Section, Baseband Section, Dual Quad Programmable Digital-Down-Converter, , Dual Quad Programmable Digital-Up-Converter, Analog-to-Digital Converter, Digital-to Analog Converter, Controller, Data Buffer, Interpolation Filter, Decimation Filter, Shaping Filter, CIC Filter, FIR Filter, Root Raised Cosine FIR Filter, Filter Compute Engine, Channel Capacity, MPSK Modulation, Intersymbol Interference.	**15. NUMBER OF PAGES** 123
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

SIGNAL SYNTHESIS WITH DYNAMICALLY-CHANGED POWER SPECTRAL DENSITY IN A SOFTWARE DEFINED RADIO TRANSMITTER

Nikolaos Apostolou
Major, Hellenic Army
B.S., Hellenic Army Academy, 1989

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN SYSTEMS ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 2003

Author: Nikolaos Apostolou

Approved by: Jovan Lebaric
Thesis Advisor

David Jenn
Second Reader

Dan C. Boger
Chairman, Department of Information Sciences

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The objective of this thesis is to synthesize signals with a dynamically changing power spectral density, in a Software Defined Radio (SDR) transmitter, utilizing the most appropriate channels, modulation schemes and transmission rates for communication, based on the noise profile (AWGN plus interferences) of the link, in order to achieve performance within some predefined acceptable levels. The objective is obtained by simulation.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to express my appreciation to the staff and faculty of the Naval Postgraduate School for making my education an unforgettable experience.

Very special thanks to Professor Jovan Lebaric for his guidance, his patience and his valuable teaching. Also, I would like to thank Professor David Jenn for his valuable guidance, support and help as my academic associate during my studies at NPS.

I want to thank my friend, Captain Georgios Zafeiropoulos, a student in the ECE Department, who with his experience in programming helped me create an ergonomic Matlab Code.

Finally this thesis is dedicated to Vasiliki, Argiroula, Georgios, and Michael who gave me the means to acquire my knowledge, to my wife Dr. Foteini Kougioumoutzaki, who supports and encourages me in every step of my life.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION TO SOFTWARE DEFINED RADIO

A. OVERVIEW

Consider the following scenario. A user of a mobile phone travels between two different global geographical regions in which the wireless technologies used are different. It is obvious that this person, in order to communicate, must carry several devices that cover the broad range of technology alternatives. This, however, is impractical. A more practical solution is to carry only one device to utilize services worldwide. This is possible only by reconfiguring the receiver to the air-interface standards, used in the respective regions, by dynamically downloading the software to cover the needed air-interface standard of a region. This flexibility is offered by Software Defined Radio (SDR) systems and helps solve problems due to differing standards and issues related to the deployment of new services and features.

B. THE DEFINITION OF SDR

The term SDR [1], [2] was introduced by Joe Mitola in 1991 to refer to the class of reprogrammable or reconfigurable radios in which the same piece of hardware can perform different functions at different times. The software radio uses a static platform while flexibility is achieved through software. A software radio is substantially defined in software with its physical layer behavior significantly altered through changes to the software. Thus, the same piece of hardware can be modified using software to tailor to the hardware desired application. The SDR is a shift from hardware intensive radios to multiband, multimode, multistandard, multiservice software-intensive radios. In other words, software modules run on a generic hardware platform consisting of Digital Signal Processors (DSPs). General purpose microprocessors are used to implement radio functions such as modulation at the transmitter and demodulation at the receiver. Before being activated, a transmitter can use channel estimation and modeling to check possible routes for transmission. A description of the modulation can be prepared and a smart antenna can be used for transmission. Software defined receivers contain the capability to measure received power and to detect the signal modulation. Adaptive technologies allow a more efficient use of the spectrum. Software radio systems are capable of down-

converting wide bands of signals because of software filtering techniques which are unrealizable with analog components.

C. ADVANTAGES AND DISADVANTAGES OF SDR

The equipment used in SDR is much smaller than the radio systems implemented using only hardware, so it can be transported more easily. SDR equipment can be used for a much wider variety of applications, in any area of the world, despite differences in communication standards. SDR systems simply have the ability to perform real-time software downloads and provide the user with the opportunity to customize settings automatically.

On the other hand, manufacturers using SDR technology, can overcome problems of upgrading systems to incorporate the latest innovations or to fix bugs, and by extension, the product for end users becomes less expensive.

This technology has drawbacks, such as higher power consumption, more processing power requirements and higher initial cost. The instruction rate of digital signal processing equipment must be of the order of 100,000 million instructions per second (MIPS). These needs have dictated areas in which research is being performed to perfect SDR system technology. The issue of a high resolution analog-to-digital conversion at an affordable price may be the final detail to be resolved.

D. LEVEL OF CONFIGURABILITY NEEDED TO QUALIFY A RADIO AS A SDR

The SDR Forum [3], in order to ensure a common view of the term SDR, uses tiers shown in Table 1, which describes various capabilities.

TIER	NAME	DESCRIPTION
0	Hardware Radio	The radio is implemented using only hardware. It cannot be modified using software. The only way for modification to occur is the replacement of hardware.
1	Software Controlled Radio (SCR)	A small number of functions can be changed using software. These functions do not contain frequency bands, modulation types, bit rates, etc.
2	Software Defined Radio	The system covers a broad frequency range (e.g. , 20-500 MHz, 1-2 GHz), and provides a variety of modulation techniques, operates in wide band or narrow band, provides communication security functions, all by executing software. Its library contains a large number of waveforms and air interfaces, and this number can be increased by adding new ones using disks or on-line load. Over the air software load is desirable but not required in the definition. In order to improve the functionality of the radio or to fix bugs, the software, should have the capability of applying new modules or replacing the old ones without reloading the entire set of software. The performance of the system is limited by the front end of the system which contains analog amplification, heterodyne mixers, and equipment which provides a lot of distortion and noise. Additionally, because the system operates in a broad frequency range, it may be necessary to switch between antennas in order to cover the entire range of the spectrum.
3	Ideal Software Defined Radio	It has the capabilities of SDR, but eliminates all the constraint factors of SDR, and improving in such way, system performance.
4	Ultimate Software Radio (USR)	This system description is used for comparison purposes only, rather than implementation. According to the Software Defined Radio Forum, it is always a lightweight component with a small current drain that can easily be incorporated into personal devices. No external antenna is required and there are no restrictions on operating frequency. It has a single connector that delivers desired information in the desired format. The connector also accepts information, uses it to modulate a signal and radiates that signal in the desired waveform or air interface. The USR accepts control information through its connector to operate and reconfigure the operating software. It can switch from one air interface format to another in milliseconds, use GPS to track the users' location, store money using smartcard technology, or provide video so that the user can watch a local broadcast station or receive a satellite transmission.

Table 1. Levels of Configurability.

E. ARCHITECTURE-DESIGN OF A SDR [1], [2]

Figure 1 depicts the various functional blocks in a generic SDR transceiver.

1. Antennas

The design of a Software Defined Radio begins with the antenna. Wideband antennas are required sequentially or in parallel to access multiple RF bands dynamically, with the characteristics of beam forming, diversity and sectorization. In this manner, interference, multipath and noise are minimized.

2. RF Front-End (RF Section)

The radio frequency (RF) section essentially consists of analog hardware modules. It is responsible for transmitting-receiving the radio frequency signal from the antenna, via a coupler and converting the RF signal into an intermediate frequency (IF) signal. The RF front end on the receive path performs RF amplification and down conversion from RF to IF. On the transmit path, the RF section performs analog up conversion and RF power amplification. It must be designed to optimize the Analog-to-Digital Converter (ADC) and the Digital-to-Analog Converter (DAC) performance. More specifically:

- The receiver must:
  - Reject undesired signals.
  - Convert the desired signal center frequency to a range compatible with ADC. The process of mixing or frequency translation may lead to undesirable non-linear distortion and introduction of additive noise.
  - Amplify the desired signal to the level required by the ADC with minimal distortion.
  - Minimize additive noise.
  - Achieve dynamic range (difference in power between the weakest and the strongest signal) compatible with that of the ADC.
- In the transmitter, the RF section converts a digital signal to analog, up converts it to the desired RF center frequency and after amplification, to the desired power level, and through filtering, the signal to be radiated. In practice, multiple stages of conversion and amplification may occur before the signal is radiated.

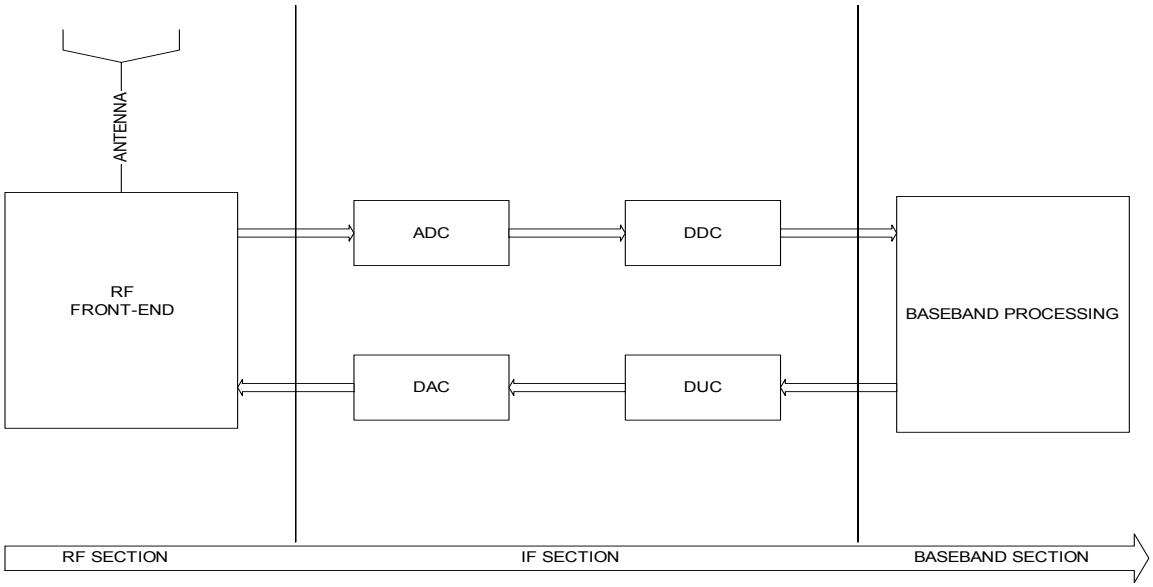

Figure 1. Block Diagram of a Generic Digital Transceiver.

3. ADC/DAC

The ADC/DAC performs analog to digital conversion on the receive path, and on the transmit path digital to analog conversion, respectively. These blocks interface between the analog and digital sections of the radio system. Usually the above conversion takes place in the IF stage. Digitizing the signal with an ADC in the IF range eliminates the last stage in the conventional model, where problems such as carrier offset and imaging, are encountered. The data converters require:

- A very high sampling rate to support wide signal bandwidths.
- A large number of effective quantization bits to support a high dynamic range.
- An operating bandwidth of several GHz in order to support the frequency translation of a signal over a greatly varying range of frequencies.
- A large spurious free dynamic range which will allow the recovery of small signals in the presence of strong interferences.
- Low power consumption.

4. Digital-Down-Converter (DDC) and the Digital-Up-Converter (DUC)

The Digital-Down-Converter (DDC) and the Digital-Up-Converter (DUC) perform digital down conversion on the receive path and digital up-conversion on the transmit path, respectively. These blocks perform modem operations, i.e., modulation of

the signal in the transmit path and demodulation of the signal in the receive path. Wideband interconnect is needed to achieve sufficient fan-out from the ADC to the digital signal processing elements. Digital filtering and sample rate conversion are often needed to interface the output of the ADC to the processing hardware to implement the receiver. The same happens in the reverse direction in the transmitter, where digital filtering and sample rate conversion are necessary to interface the digital hardware to the DAC that converts the modulated waveform to an analog waveform.

5. Baseband Section

The baseband section performs baseband operations, such as connection setup, equalization, frequency hopping, timing recovery, and correlation, and also implements the link layer protocol which is layer 2 in the Open Systems Interconnection (OSI) protocol model [1]. The DDC/DUC and the baseband processing operations require large computing power and these modules are generally implemented using DSPs, field programmable gate arrays (FPGAs) or application specific integrated circuits (ASICs). The DSPs use microprocessor-based architectures and support programming in high level languages such as C. They are flexible because they can be programmed repeatedly even with a high level language (HLL) such as C. Modifications and upgrades can be made via this HLL reducing the design time for each iteration. The ASICs implement the system circuitry in fixed silicon, resulting in the most optimized implementation in terms of speed and power consumption. This design requires a sophisticated circuit design. The FPGAs allow more flexibility than an ASIC but less than a DSP. They provide much hardware–level configurability. Lastly, the software design must be based on an object-oriented approach.

F. GENERAL CONCLUSIONS [1]

The main characteristic of SDR is its dynamic adaptation to the radio environment through the reconfigurability of its components. Simply changing the software that runs the DSP and its installation completely changes the radio interface. This characteristic provides SDR systems the ability to support a wide range of mobile radio standards. Present technologies do not have suitable solutions for implementing conventional cost effective SDR systems. It is most likely that solutions will be proposed in the near future.

G. OVERVIEW

Chapter II describes the main functions of the card "Model 253 Waverunner plus PCI." This card is a polychannel (8 channels) programmable digital SDR transceiver.

Chapter III discusses the concept of "Water Pouring" in the frequency domain. Given the total transmitted power and the attenuation-to-noise ratio for a communication channel, suitable allocation of the transmitted power in the lower noise areas of the channel, allows the transmission of high information rates. MATLAB CODE 1 describes a way of finding the noise profile (AWGN and random interferences), and the lower noise areas in the operating bandwidth of the transmitter of the card.

In Chapter IV covers the simulation of the basic functions of the card and its performance with AWGN (MATLAB CODE 2) under various assumptions.

In Chapter V, diagrams are extracted which describe the performance of the card in the presence of AWGN and interference of different power (MATLAB CODE 3). Additionally, by transmitted power control, using the most appropriate MPSK modulation scheme with $k = 2,\ 3,\ 4$, the most appropriate number m of the eight channels of the card, and the m lower noise spectral areas in the operating bandwidth of the transmitter, the maximum m parallel transmission bit rate is achieved (MATLAB CODE 4). In this way, the concept of "Water Pouring" in frequency domain is applied, in the case of a SDR transceiver.

Chapter VI describes the obtained conclusions from the simulations.

The appendix finally lists the four matlab codes referred to above.

THIS PAGE INTENTIONALLY LEFT BLANK

II. DESCRIPTION OF THE CARD "MODEL 253 WAVERUNNER PLUS PCI" OF RED-RIVER COMPANY

A. GENERALLY

The Model 253 Waverunner Plus PCI is a polychannel programmable digital transceiver (Figure 2), which adds a high performance software defined radio capability to any computer with a PCI back plane. It provides a high degree of flexibility and precision through the use of programmable digital filters, sample rates and gain control functions. A general description of the main parts and functions of the card is provided below. A more detailed description of the card can be found in [4], [5], [6], [7].

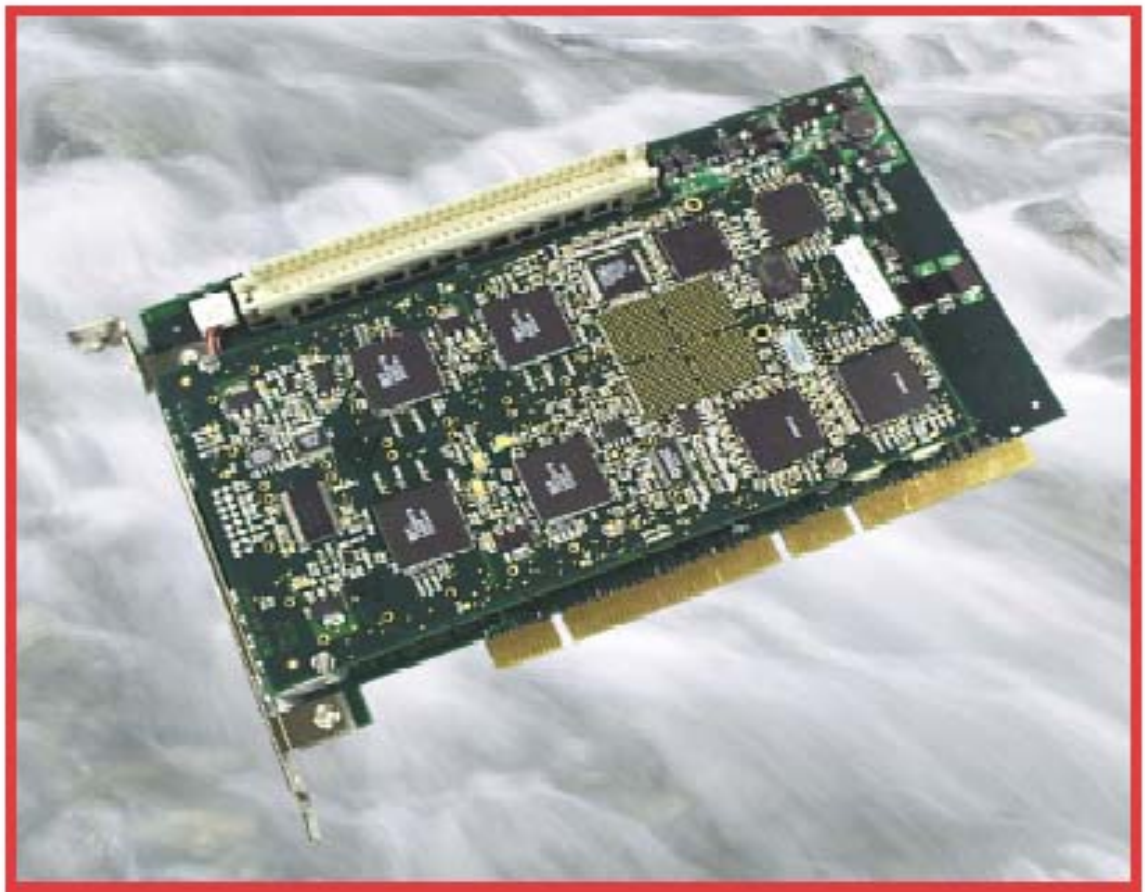

Figure 2. WaveRunner Plus PCI, Model 253 (From: [5]).

B. DESCRIPTION OF THE TRANSMITTER

1. Basic Parameters

The operating frequency of the transmitter is in the 3 to 40 MHz range. Its sample clock rate is 93 MHz, which is determined by an internal on board phase-locked loop (PLL) synthesizer. The same PLL synthesizer determines the receiver's sample clock. The A/D and D/A converter sample clock can also be supplied directly from an external source through the J2, J3 connectors shown in Figure 3.

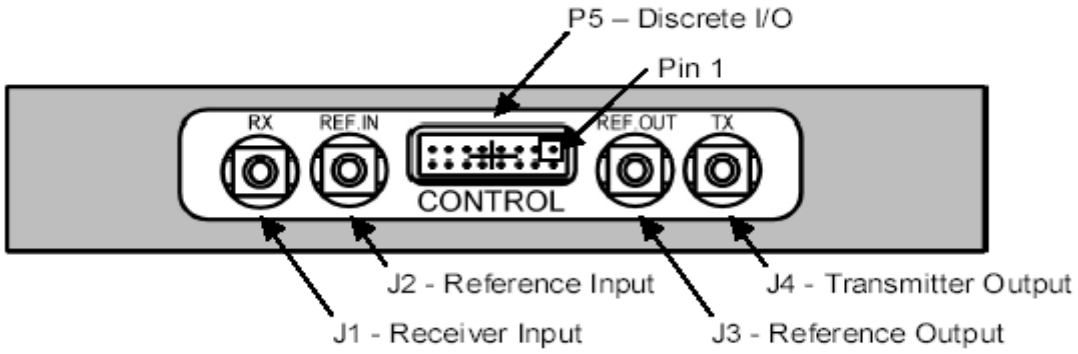

Figure 3. Front Panel Connectors of the Card (From: [5]).

2. Main Sections

The transmitter consists of the five sections listed below:

- Controller
- Transmitter data buffer
- Dual Quad Programmable Digital Up Converter (QPUC)
- D/A Converter
- Transmitter front end

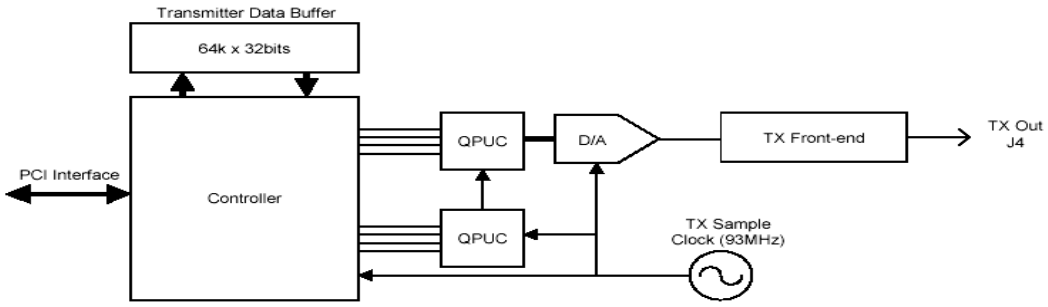

Figure 4. Transmitter Main Sections (From: [5]).

a. Controller

The controller performs all local commands and control functions while it also acts as an interface to the host computer. The interface supports 32 bit transactions operating at 33 MHz.

b. Transmitter Data Buffer

Digital data samples are sourced through the PCI interface and stored in the buffer using DMA transfer[1]. The capacity of the buffer is 256 Kbytes, capable of storing 64 KSamples of complex data (16-bit I, 16-bit Q). It acts as a FIFO, performing data rate translation between the PCI bus and the QPUC inputs. By writing configuration data to specific registers, it is possible to:

- Split the buffer's memory from one to eight areas of different sizes. Each memory area can be assigned to a different channel in the dual QPUC.
- Initialize read-write pointers.
- Define the memory, start and stop addresses.
- Define the memory area threshold which specifies how much data must be received into the memory area before an interrupt is generated indicating that the memory area is ready to be processed.

c. Quad Programmable Up Converter (Model Intersil ISL 5217)

Each one has four channels for a total of 4x2=8 channels. Each QPUC channel consists of the following parts (Figure 5):

(1) FIFO Holding Registers. These provide the interface and data storage between the input source and the shaping filter. Its input is I, Q samples of 16 bits in the parallel or serial mode.

(2) Shaping Filter. This provides the necessary pulse shaping required on the input data to implement various modulations. There are three modulation modes. Mode 00-QASK configures the QPUC as a BPSK, OQPSK, MSK or m-QAM modulator. Mode 01-FM, with a band limiting filter, configures the QPUC as an FM

[1] Direct Memory Access (DMA). It is the transferring of data from one storage device, to another, without using the CPU. It is faster than the programmable Input /Output method, used for the transfer of data the CPU.

modulator with post-modulation filtering and mode 10-FM, with pulse shaping, configures the QPUC as an FM modulator with pre-modulation base-band pulse shaping.

The shaping filter is a user-configured FIR filter with up to 256 taps. It is a low-pass implementation filter that provides sufficient rejection to suppress images generated by subsequent interpolation stages.

(3) FM Modulator. It is referred to as FM modulation.

(4) Gain Profile. It is unavailable in this specific model of the card.

(5) Half Band Filter. A filter with fixed coefficients and an interpolation rate equal to 2. Its default mode is "Bypass."

(6) Interpolation Filter. It resamples the shaped I, Q data to establish the final output sample rate of the channel, which is **always** 93 MHz. The interpolation factor (L) is from 4 to 4096.

(7) Timing NCO. It controls the processing of the shaping filter and the processing of the interpolation filter.

(8) Carrier NCO-Complex Mixer. It multiplies in the two mixers, the I and Q interpolated sample data, coming from the interpolation filter, with the sine or cosine generated by the carrier NCO. The mixers can be bypassed making the carrier frequency equal to zero.

(9) Gain Control. Provides attenuation of the transmitted output over a range from 0 to 144 dB.

(10) Channel Summer. The four QPUC channel outputs are combined in the channel summer, along with the cascaded QPUC output, to create a real eight channel maximum composite data stream for the D/A converter.

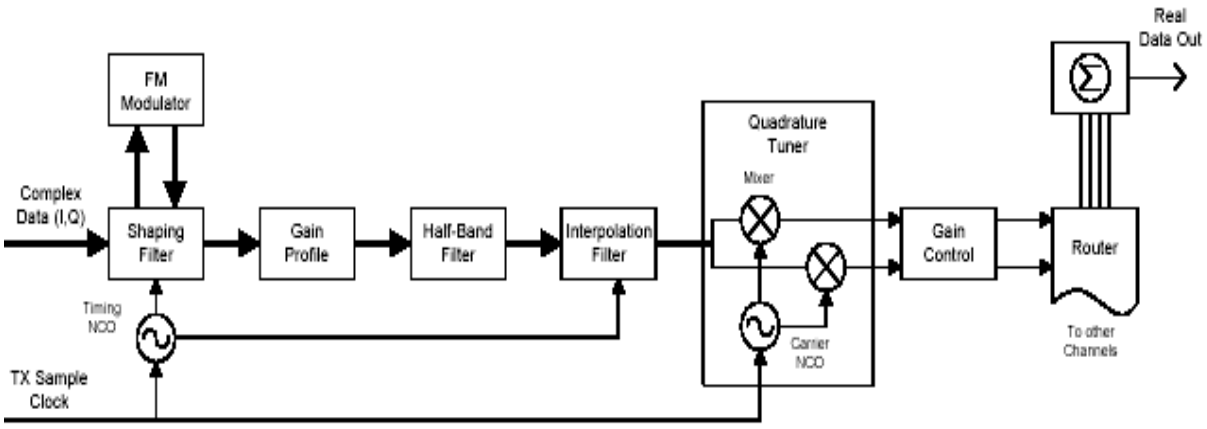

Figure 5. Single QPUC Channel (From: [5]).

d. D/A Converter

It is driven directly by the QPUC digital output (Figure 4) to generate an analog signal. Its resolution is 14 bits and the sampling rate is 93 Msps.

e. Transmitter Front End

The D/A output passes through a 9-pole Chebyshev 40 MHz low pass reconstruction filter prior to the final amplifier. The Tx Output, (connector J4 in Figures 3 and 6), through a 50 Ohm cable connected to the antenna.

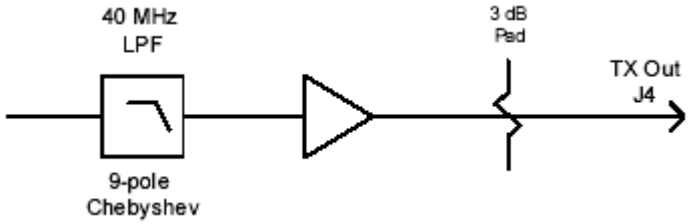

Figure 6. Transmitter Front End (From: [5]).

C. DESCRIPTION OF THE RECEIVER

1. Basic Parameters

Its frequency range is from 0.1 to 40 MHz with a sample clock rate of 93 MHz.

2. Main Sections

The receiver as well as the transmitter consists of five sections as shown in Figure 7:

- Receiver front end
- A/D Converter
- Dual Quad Programmable Digital Down Converter (QPDC)
- Receiver Data Buffer
- Controller

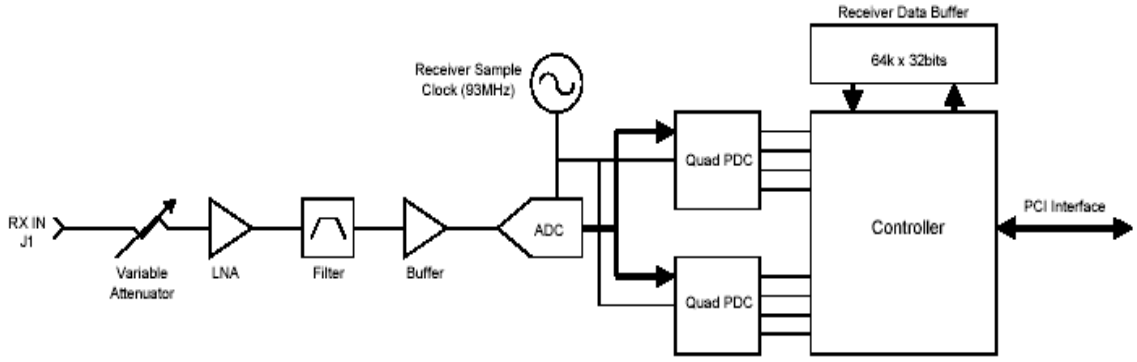

Figure 7. Receiver Main Sections (From: [5]).

a. Receiver Front End

The received signal enters the card through the receiver input, (connector J1, Figures 3 and 7). The function of this part of the receiver is to amplify and filter the analog input signal in preparation for A/D conversion. The front end consists of a digitally-controlled attenuator, a low-noise amplifier (LNA), and a filter element which is a 7-pole Chebyshev Low-pass filter with a pass-band frequency at 40 MHz.

b. A/D Converter

The received signal is sampled using an A/D converter operating at a fixed rate of 93 MHz with a resolution of 14 bits.

c. Quad Programmable Down Converter (Model Intersil ISL 5216)

The dual QPDC provides up to eight independent data channels. It accepts raw sample data directly from the A/D converter and performs the computationally intensive tasks of tuning, filtering, decimation, gain control and re-sampling. The digital

down converter is one of the key receiver signal conditioning components in a software radio system. The QPDC consists of the following parts (Figure 8):

(1) Input Level Detector. It monitors the signal level as it enters the QPDC. This signal level is useful for controlling gain/attenuation blocks ahead of the A/D converter.

(2) Carrier NCO/ Quadrature Tuner. The tuning function is implemented using a NCO and quadrature mixer. Here the samples (I, Q) are multiplied by quadrature sine waves.

(3) CIC Filter. Next, the signal is filtered by a cascaded integrator/comb (CIC) filter. The CIC filter is an efficient architecture for decimation filtering. Its transfer function is given by the equation:

$$H(f) = \left(\frac{\sin(\pi \cdot M \cdot f)}{\sin(\pi \cdot f / R)}\right)^N \tag{1}$$

where

M =number of delays which generally is 1 or 2 and in this application is 1
N =Number of stages
R =Decimation factor.

It is a coarse filter which is used to extract the band of interest and decimate the signal prior to it passing from the precise filters of the filter compute engine (FCE). Its minimum decimation factor is R=4 and its gain is R^N.

For decimation up to R=512, N=5 is used. Other corresponding values of (R, N) are (2048,4), (32768,3), and (65536,1) or (65536,2).

(4) Filter Compute Engine (FCE). Its objective is to fine tune the processed signal spectrum created by the CIC filter. The FCE provides a variety of user configurable filters. The user can implement a single FIR or a combination of filters each with different decimation factors.

(5) AGC. A full featured AGC loop is built into the QPDC for high performance signal level control.

(6) Cartesian to Polar Converter. This converter converts data from the complex I,Q format to a polar form of magnitude and phase. Both data formats are available to the user to save software overhead in converting between the two systems.

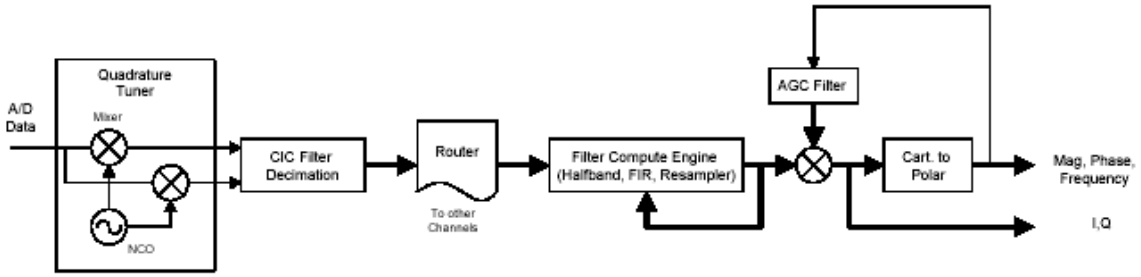

Figure 8. Single QPDC Channel (From: [5]).

d. Receiver Data Buffer

Its capacity and the way it is configured is the same as in the transmitter. It acts as a FIFO to perform data rate translation between the QPDC outputs and the PCI bus.

e. Controller

It is the same as in the transmitter. In Figure 9, the entire block diagram of the card is shown.

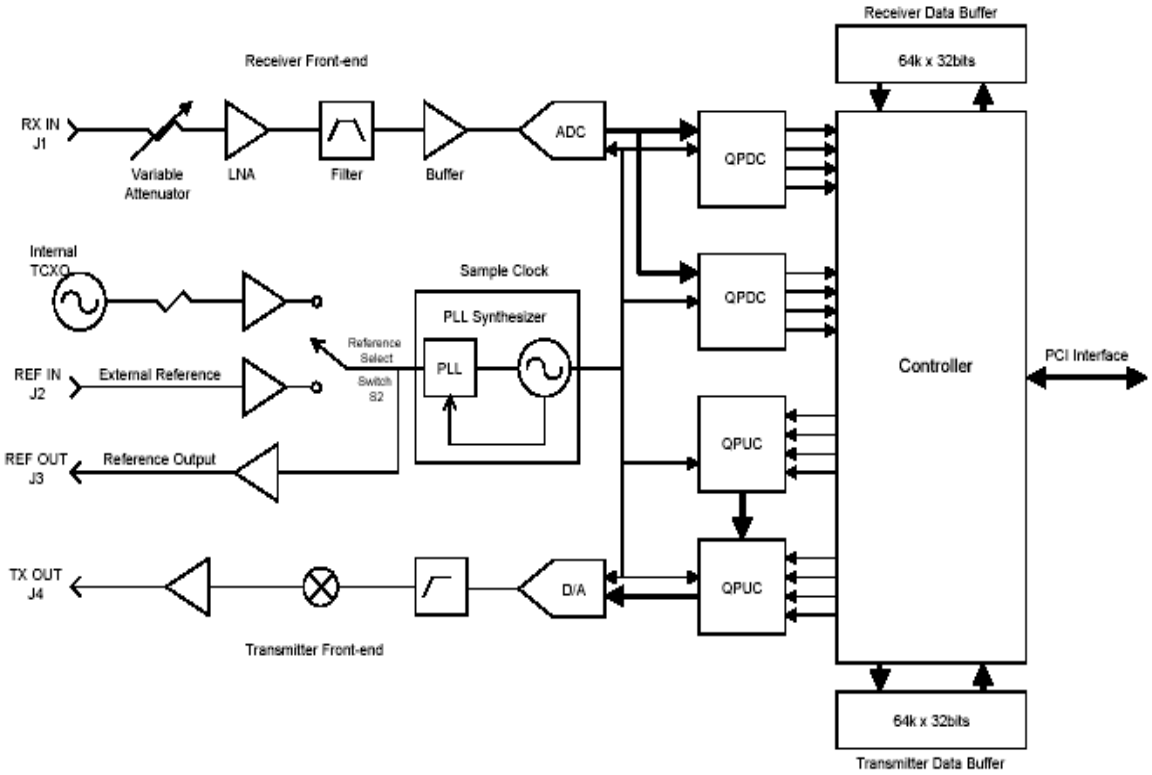

Figure 9. Wave Runner Plus Detailed Diagram (From: [5]).

THIS PAGE INTENTIONALLY LEFT BLANK

III. "WATER POURING" IN THE FREQUENCY DOMAIN

A. CHANNEL CAPACITY THEOREM

Channel capacity is the maximum bit rate at which reliable communications can occur. Given a band limited transmission channel with additive white Gaussian noise (AWGN), the capacity is described by the equation

$$C = W \cdot \log_2(SNR + 1) \tag{2}$$

where:

C is the capacity of the channel in bps,

W is the noise equivalent bandwidth in Hz,

SNR is the received signal to noise ratio.

Shannon's noisy channel coding theorem states that if $C \geq Rb$ where Rb is the transmitted bit rate of a signal over a channel, in principle, error free communication is possible. Also, at any transmission rate above C bps, i.e., for $Rb \geq C$, no transmission method can be devised that will eliminate all errors.

The channel capacity theorem as expressed above is limited by two factors. Its bandwidth has the form of the frequency response of an ideal bandpass filter (BPF) and the noise is completely white. Thus, the theorem must be expanded to more arbitrary channels with arbitrary stationary noise.

If a sufficiently small region of the channel spectrum is taken, i.e. suppose δf, then it can be assumed that the attenuation vs. frequency is approximately constant and the spectrum of noise is approximately flat. Thus, for this small spectral interval the channel capacity theorem holds. If separate information rates are added (predicted by the capacity theorem for all previous small regions of spectrum δf) an approximate prediction can be made for the information rate for the entire spectrum. These ideas can be expressed by the following equation,

$$C = \sum_f \log_2[SNR(f) + 1]\delta f \tag{3}$$

or for regions δf infinitesimally small, the sum becomes the integral,

$$C = \int_f \log_2[SNR(f)+1]df \tag{4}$$

where $SNR(f)$ is the received signal-to-noise ratio in the vicinity of frequency f.

From arguments that lead to the capacity theorem, it is obvious that the *SNR* in the theorem is to be measured at the receiver, where decisions must be made. It is not sufficient to specify transmitted power at the frequency of interest $P(f)$ since for each small spectral region, it is the transmitted power times channel attenuation $H(f)$ that must be compared to noise power $N(f)$ at that frequency. In other words, the received *SNR* is given by,

$$SNR(f) = \frac{P(f)\cdot H(f)}{N(f)} \tag{5}$$

and the total information rate becomes,

$$C = \int_f \log_2[\frac{P(f)\cdot H(f)}{N(f)}+1]df \tag{6}$$

with total transmitted power of,

$$P = \int_f P(f)df. \tag{7}$$

For maximal capacity, it is necessary to know the optimal transmitter power distribution $P(f)$. The entire transmitter power must not be spread equally across the entire spectrum. More transmitted power should be spent where the ratio $\frac{N(f)}{H(f)}$ is lower, so the received SNR will be higher. Thus, the power should be distributed according to the relations,

$$P(f) = Q - \frac{N(f)}{H(f)} \quad \text{for } \frac{N(f)}{H(f)} < Q$$

and $$P(f) = 0 \qquad \text{for } \frac{N(f)}{H(f)} > Q \tag{8}$$

where the value of Q is determined by the requirement of equation (6) where the total power P.

B. "WATER POURING CRITERION"

Robert Gallager [8] created the "water pouring criterion." To understand this term, attenuation may be equated to noise distribution ratio as a bowl with an irregular bottom, and the total amount of power to be transmitted as the amount of water in a glass poured into the bowl. No water covers the areas where the bowl's bottom is too high. Where the bowl's bottom is low, the height of the water is maximal.

Similarly, with "water pouring" in the frequency domain, given the total transmitted power and attenuation-to-noise ratio, we "pour" power using equation (7) in order to find the power distribution of the signal which will give the highest information transfer rate.

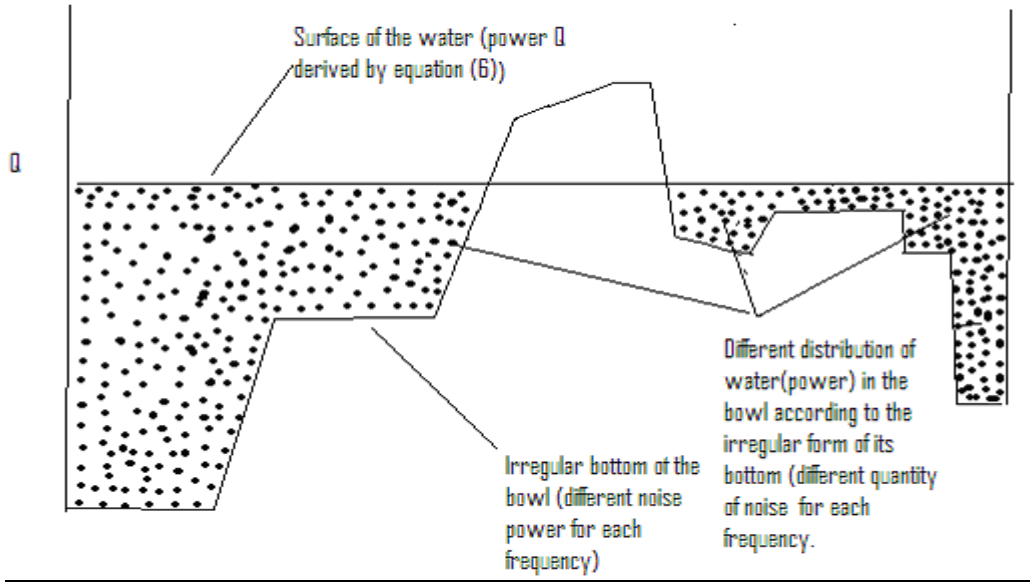

Figure 10. Representation of the Water Pouring in the Frequency Domain using a Bowl with an Irregular Bottom.

C. DEFINITION OF THE LOWER NOISY SPECTRAL AREAS IN THE OPERATING BANDWIDTH OF THE TRANSMITTER OF THE RED RIVER CARD

From the above theoretical approach, the operation of the transceiver of the Red River Card is described.

- The receiver must decide details of the noise profile of the communication channel and inform the transmitter.
- The transmitter, using this information, must define how it allocates transmitted power among the lower noise spectral areas of the communication channel and, what modulation scheme and transmission rate will be used in order to achieve a performance within some predefined acceptable level. The above approach does not have a unique solution and varies each time assumptions are changed. Initially the task is to find the lower noise spectral areas in the operating bandwidth of the transmitter.

The operating bandwidth of the transmitter extends from 3-40 MHz, the sampling rate is equal to 93 MHz and the dual QPUC can provide up to 8 channels, based on the card description.

To find the *n*th lower noise spectral areas MATLAB, AWGN and two BPSK signals are used to represent interference signals, in the operating bandwidth of the transmitter. The lower noise spectral areas are directly related with the QPUC channels. These areas, in the future, will be covered by signals from the QPUC channels. Their numbers extend from 1 to 8 while their bandwidth must not exceed the maximum bandwidth of a QPUC channel. The code assumes that all the lower noise spectral areas have the same bandwidth, bw, each time. Figures 11 through 13 provide results of the simulation.

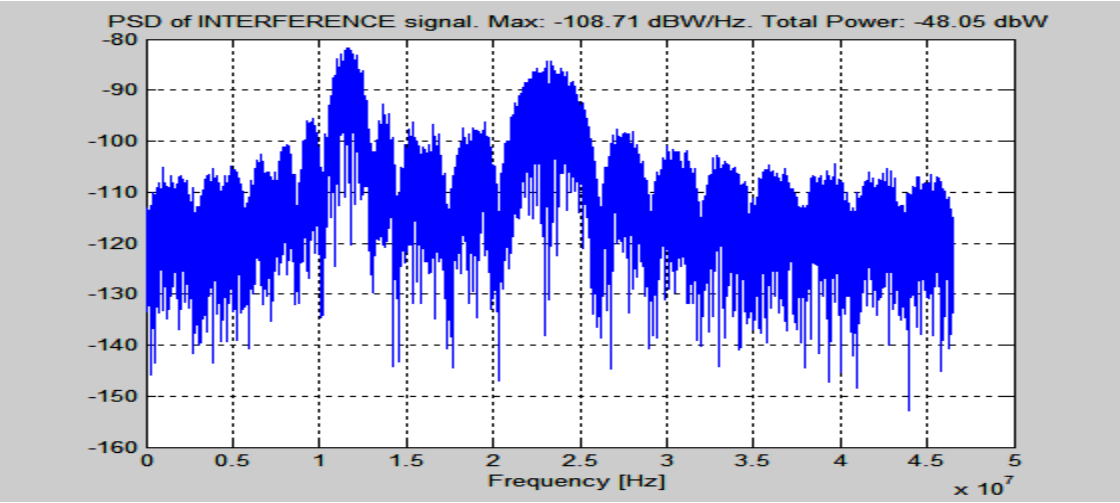

Figure 11. Operating Bandwidth of Transmitter Affected by AWGN and Two Interferences, (BPSK Signals which are at $fc_1 = 23.25MHz, fc_2 = fc_1 / 2$).

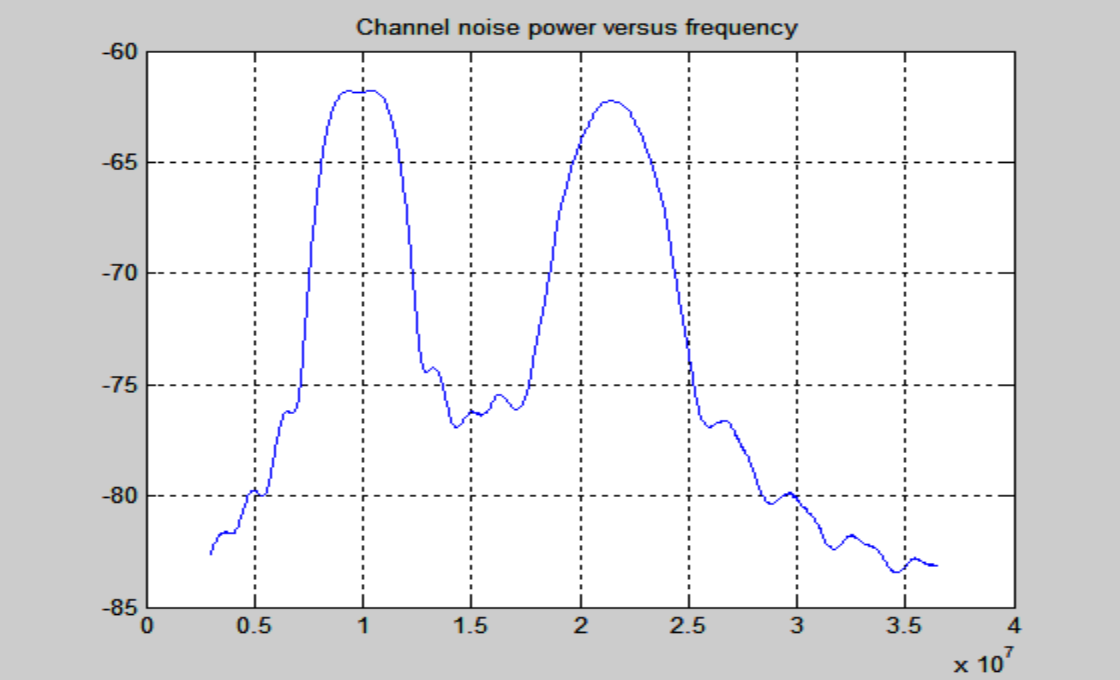

Figure 12. Noise Fluctuation for Each Channel throughout the Operating Bandwidth of the Transmitter. (Detailed explanation about the derivation of this scheme exists in Matlab Code 1)

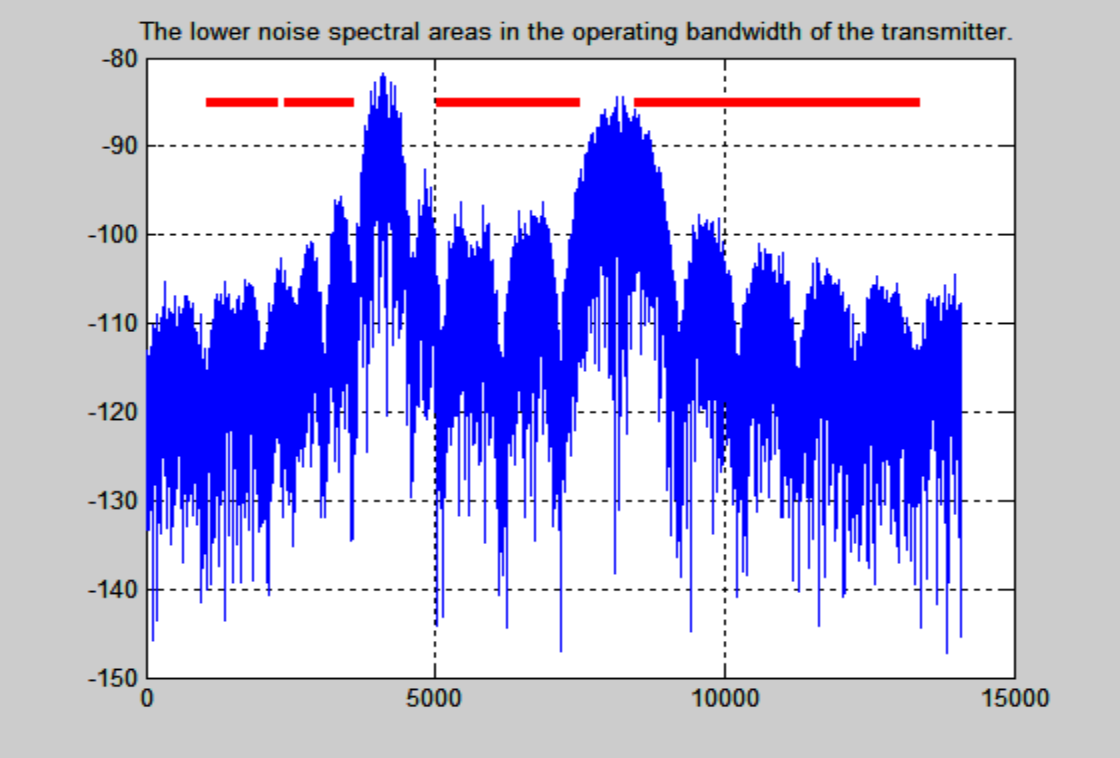

Figure 13. Red Lines Indicate the Lower Noise Areas of the Operating Bandwidth of Transmitter. The Bandwidth of Each Area is equal to 3.5 MHz.

Although the red lines in the above plot denote the lower noise areas, they do not indicate what the noise levels are and, do not specify the exact initial frequency. In future uses, carrier frequencies will be needed for transmitted signals inside the areas. The Matlab variables QNoisepower and frequencyareas provide these answers.

4.530e-9	5.505e-9	7.174e-9	1.621e-8	2.025e-8	2.36e-8	3.906e-8	1.652e-7

Table 2. Lower Noise Spectral Areas in the Transmitter Operating Bandwidth, in Ascending Order (from Variable QNoisepower). Units are Hz.

3.454e+7	3.0027e+6	3.1035e+7	2.7533e+7	1.431e+7	6.786e+6	1.78121e+7	2.403e+7

Table 3. Initial Frequencies of the Above Lower Noise Spectral Areas (from Variable Frequency areas). Units are Hz.

Of course, the desired number n, of the lower noise areas and their bandwidth can be changed. Thus, for 7 spectral areas with a bandwidth of each equal to 2.5 MHz and for interference signals at 31 MHz and 6 MHz in the presence of AWGN, see Figure 14.

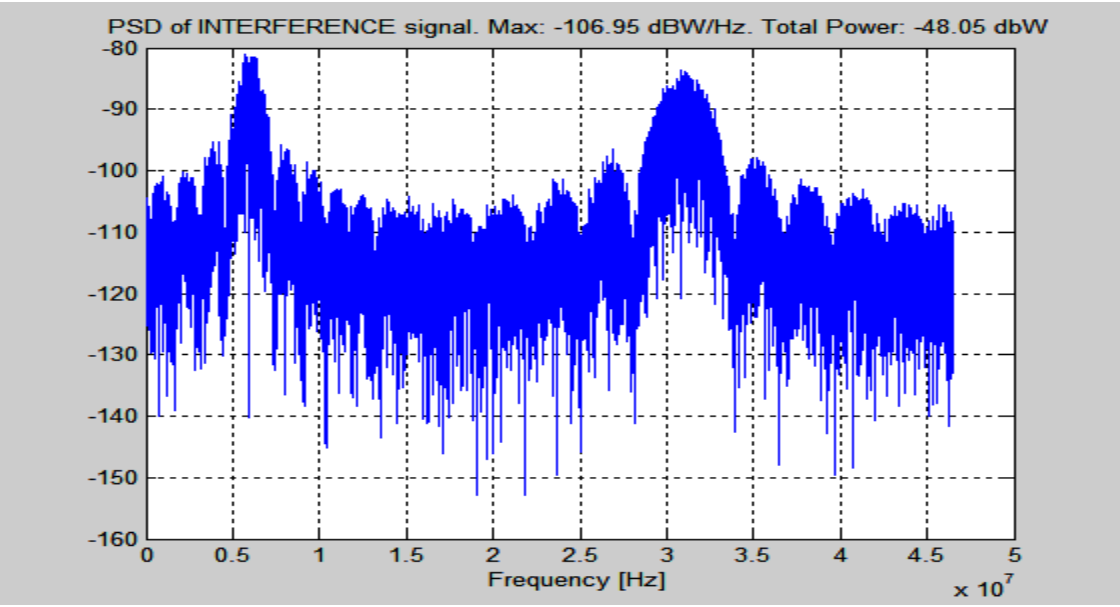

Figure 14. AWGN and Interference BPSK Signals at Frequencies $fc_1 = 6MHz, fc_2 = 31MHz$.

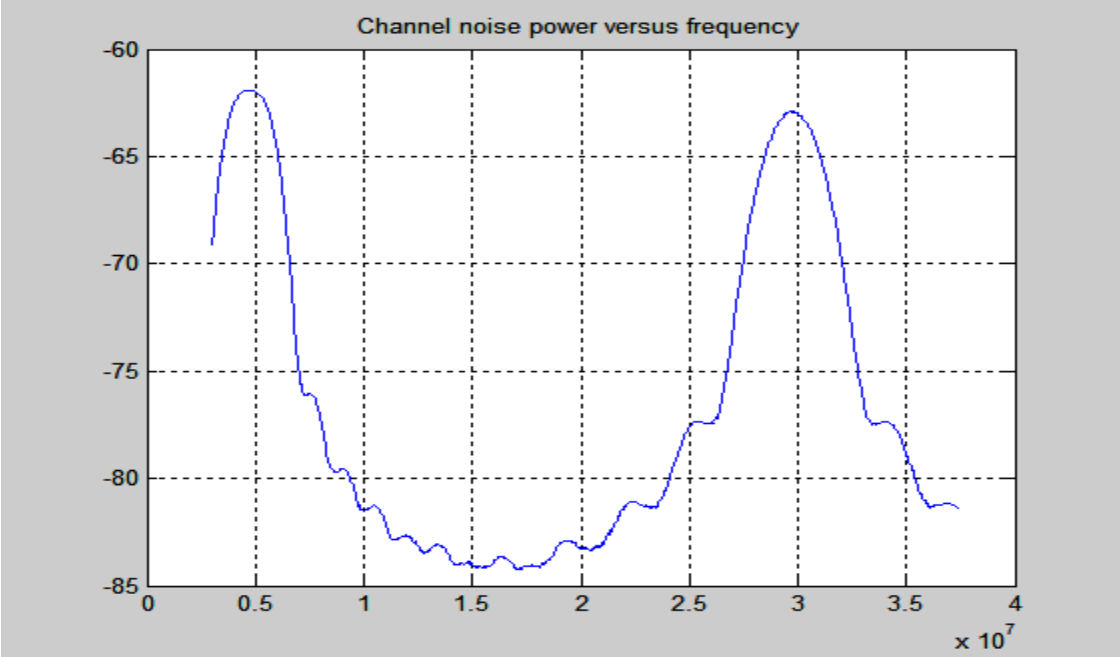

Figure 15. Noise Fluctuation for Channels in the Transmitter Operating Bandwidth.

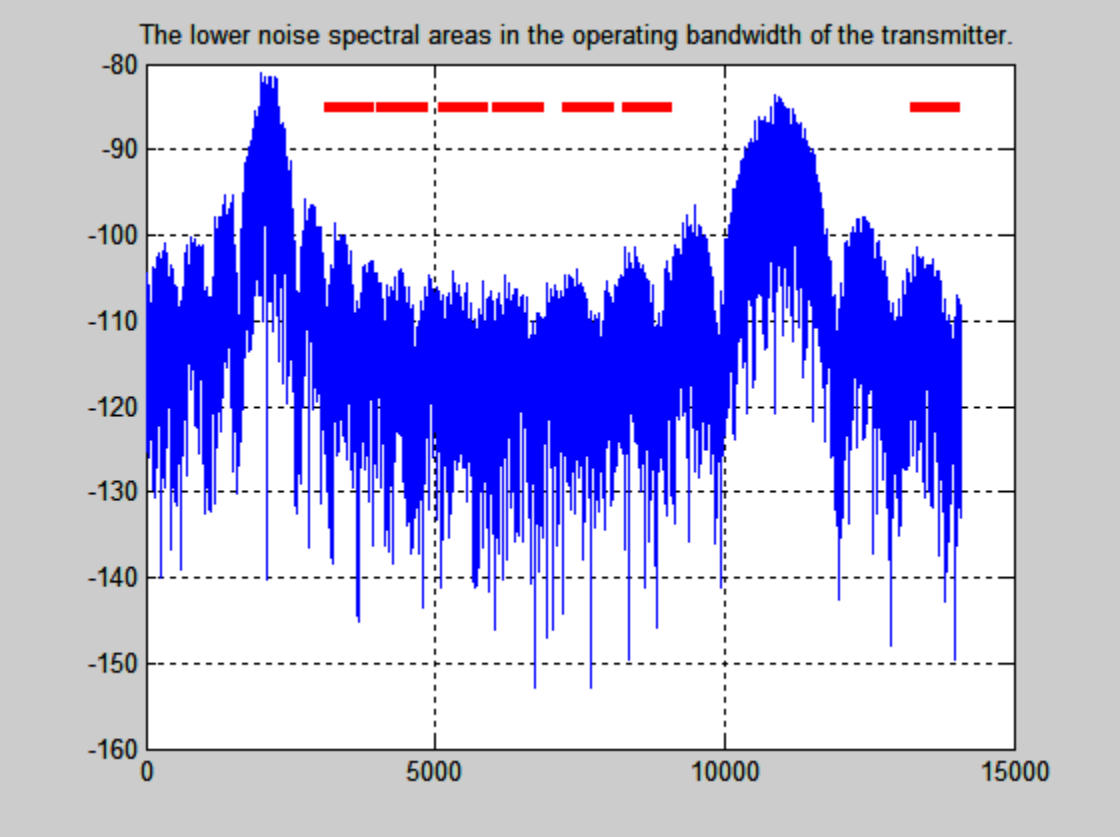

Figure 16. Red Lines Indicate the Lower Noise Areas of the Transmitter Operating Bandwidth.

3.773e-9	3.9373e-9	4.5987e-9	5.1642e-9	7.1993e-9	7.2346e-9	1.097e-8

Table 4. Lower Noise Spectral Areas in the Transmitter Operating Bandwidth, in Ascending Order (from Variable QNoisepower). Units are Hz.

1.7049e+7	1.4347e+7	2.0466e+7	1.1378e+7	3.7487e+7	2.3392e+7	8.767e+6

Table 5. Initial Frequencies of the Above Lower Noise Spectral Areas (from Variable Frequencyareas). Units are Hz.

IV. SIMULATION OF BASIC FUNCTIONS OF THE RED-RIVER CARD IN MATLAB AND CARD PERFORMANCE WITH AWGN

A. MATLAB SIMULATION FOR THE BASIC FUNCTIONS OF THE RED RIVER CARD

The performance of the card is studied, with AWGN present, by simulating its main functions in Matlab (MATLAB CODE 2). The modulation schemes used are QPSK, 8-PSK and 16-PSK. The probability of error, Pb, is fixed and equal to 10^{-5}, which is a representative BER for modern communication systems. The card simulation is described below and refers to a single QPUC-QPDC channel.

In the transmitter:

- Creation of complex data (I, Q) at an initial sampling frequency, $f_{sinitial} = 93MHz/4 = 23.25MHz$ and for a symbol rate $R_s = 1.453125Msps$. Below, how the symbol rate can be determined will be examined as well as investigating how its changes affect the performance of the system.
- Baseband modulation of I, Q data (Figure 17).
- Interpolation[2] of baseband data using the smallest interpolation factor provided by the Interpolation Filter of the card, $L = 4$, in order to establish the final output sample rate which is always equal to the clock rate, i.e. $f_{sclock} = 93MHz$. For this reason, we start with $f_{sinitial} = \frac{f_s}{4}$.

[2] According to [4], in the transmitter, the data FIFO outputs are routed to the shaping filter where they are interpolated by a factor 4, 8 or 16 and shaped using a FIR filter with up to 256 taps. Afterwards, the data shaping filter outputs are passed through the interpolation filter, which interpolates them by a suitable factor, such as its output, to provide data at the clock frequency. Considering the difficulty in obtaining the details of this interpolation filter, from Intersil, in the simulation, this filter is disregarded, assuming that the interpolation to the final fsclock, which is equal to the sample clock rate, is done by the shaping filter using L=4 as the interpolation factor. The opinion of the manufacturer of the card (Red River) is that this approach provides a good model in order to investigate the system's performance.

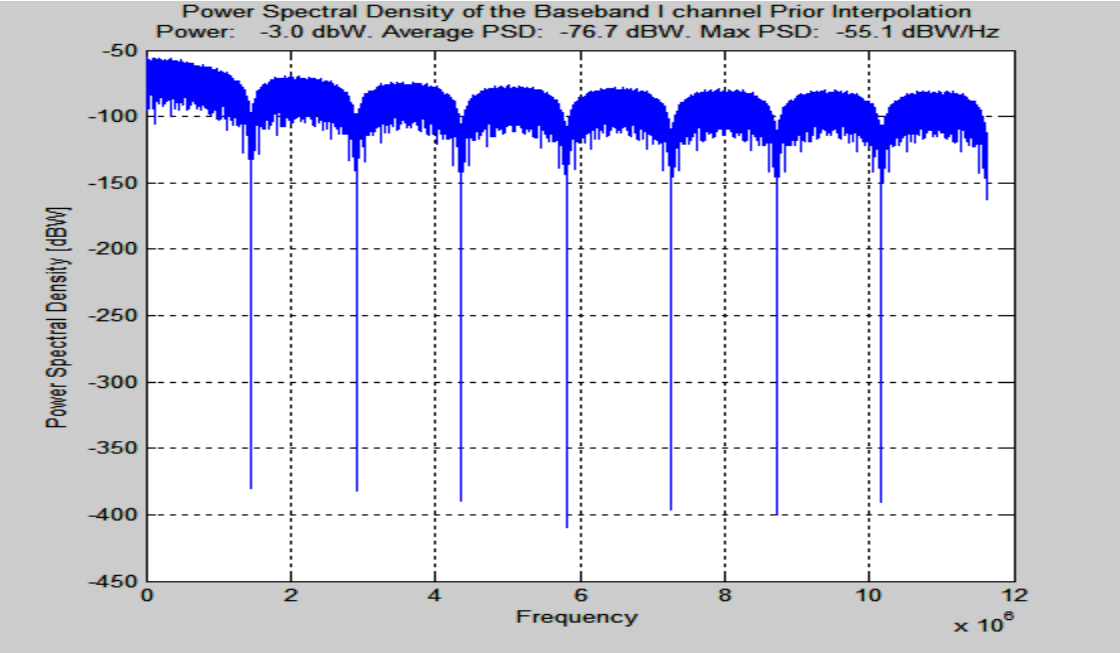

Figure 17. PSD of the Baseband I Channel Prior Interpolation. (QPSK Modulation, $R_s = 1.453125 Msps$).

- Based on the theory of digital filters, after interpolation, a suitable filter must be used to remove the image frequencies created by up sampling. The filter cutoff frequency must be $\frac{\pi}{L} = \frac{\pi}{4}$. A simple FIR filter is defined which represents the FIR of the card's shaping filter with the analog cutoff frequency as,

$$\omega = \frac{\pi}{L} = \frac{\pi}{4} = 2 \cdot \pi \cdot \frac{f_c}{f_{sclock}} \Rightarrow f_c = \frac{f_{sclock}}{8} = 1.625 MHz \quad (9)$$

and, pass band frequency $\frac{f_{sclock}}{16} = 5.8125 MHz$. The number of required taps, according to MATLAB CODE 2, is 32. Figure 18 shows the waveform of the I signal after interpolation and filtering using the above FIR filter. Figure 19 is the transfer function of this filter.

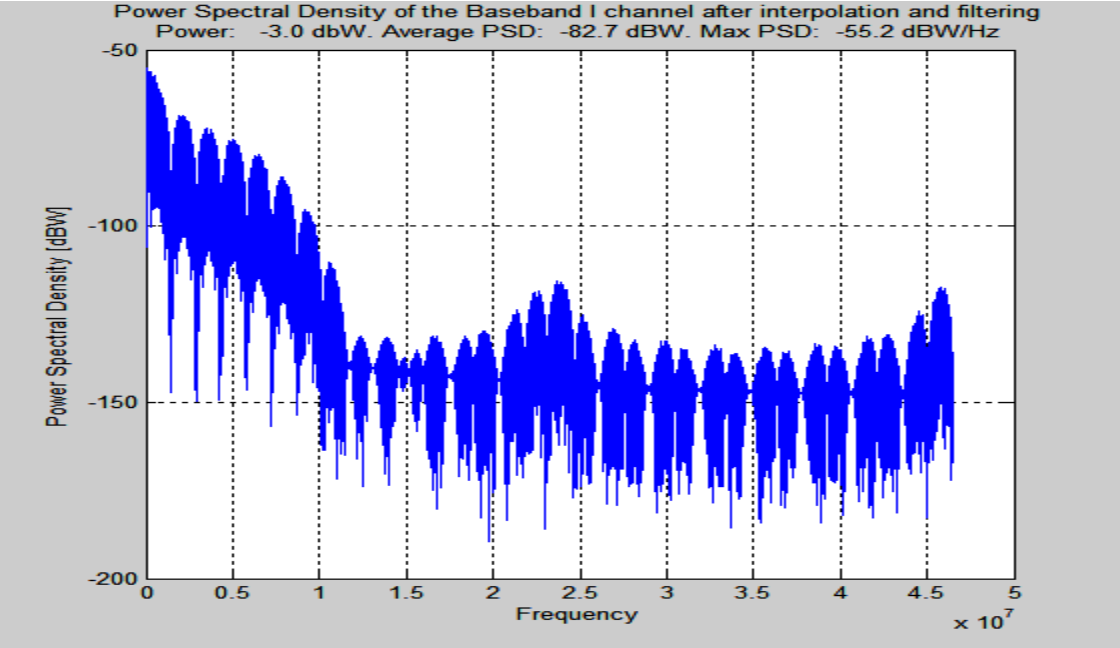

Figure 18. PSD of I Baseband Signal after Interpolation and Filtering using a FIR Filter.

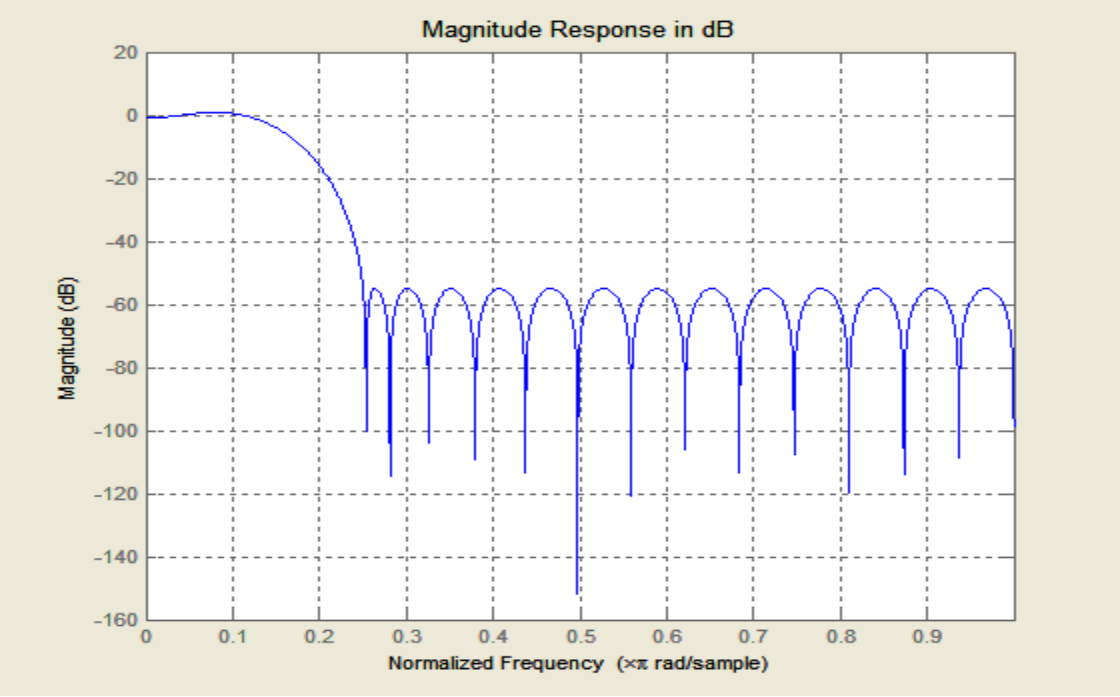

Figure 19. Transfer Function of FIR Filter Used After Interpolation.

- After interpolation, modulation and transmission of the signal take place at the desired carrier frequency. Figure 20 illustrates the transmission of the interpolated I, Q data at a carrier frequency of $10 MHz$.

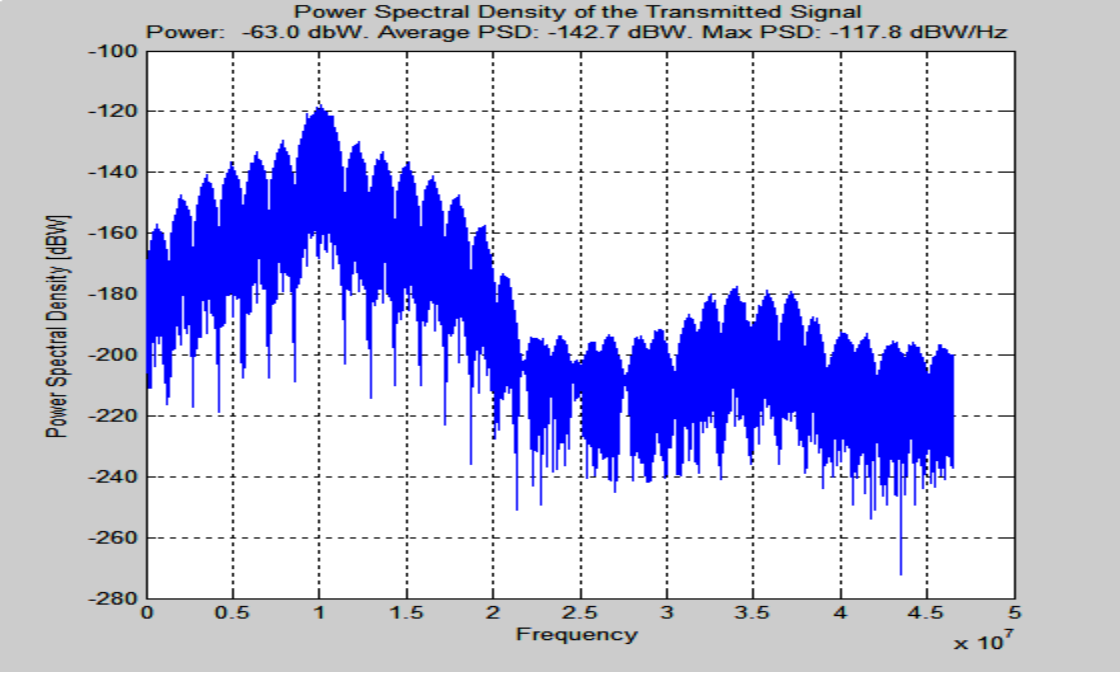

Figure 20. Transmission of the Modulated Signal.

The receiver provides:

- Demodulation of the received signal affected by AWGN to the baseband. (To make the signal visible, AWGN was omitted in the figures below.)

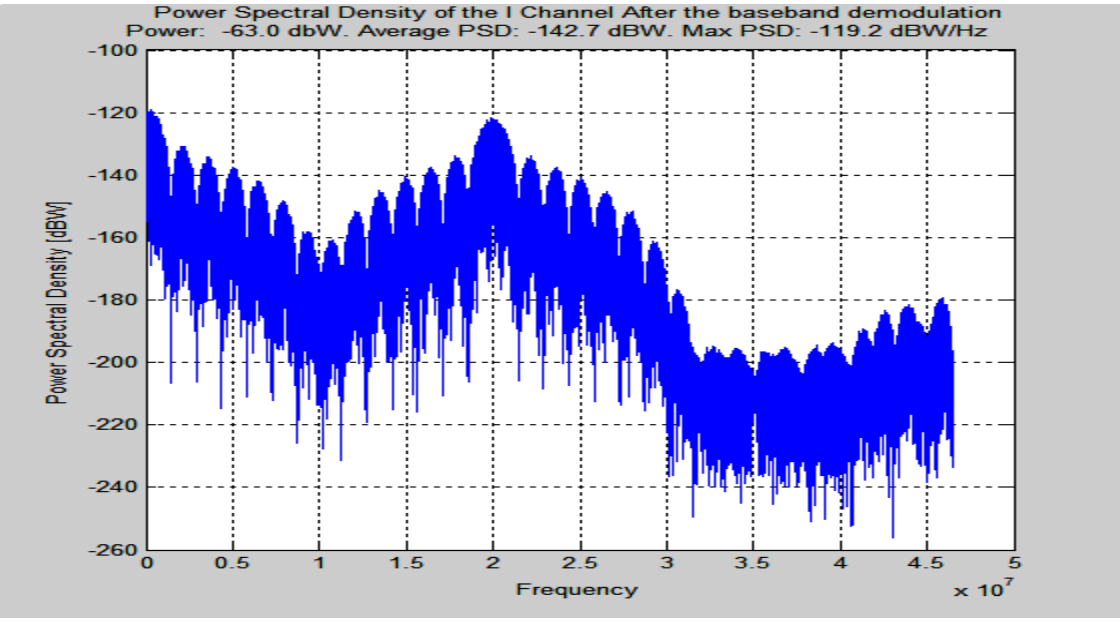

Figure 21. PSD of Demodulated I Signal in the Receiver.

- Decimation of the baseband signal is accomplished in two stages. In the first stage, a CIC filter is used with the smallest decimation factor R_{CIC}=4, number of delays M=1 (Red River's default value) and number of stages N=5 (Red River's default value for decimation factors 4 to 512). The CIC filter is used to extract the band of interest and drop the sample rate prior to precision filtering from the FCE section. In the second stage, a simple FIR filter, representing the filter in FCE, with a decimation factor equal to R_{FCE}=4, is used. Based on the theory of digital filters, it is known that prior to decimation, it is necessary to filter out all frequencies from the signal's spectrum above $\frac{\pi}{R_{FCE}} = \frac{\pi}{4}$, in order to avoid aliasing after the down-sampling. The digital cutoff frequency for this filter must be $\frac{\pi}{R_{FCE}} = \frac{\pi}{4}$. A FIR filter is defined with an analog cutoff frequency of

$$\omega = \frac{\pi}{R_{FCE}} = \frac{\pi}{4} = 2 \cdot \pi \cdot \frac{f_c}{f_{safterCICFilter}} \Rightarrow$$
$$f_c = \frac{f_{safterCICFilter}}{8} = \frac{23.25}{8} \cong 2.9MHz \quad (10)$$

and pass band frequency $\frac{f_{safterCICFilter}}{16} \cong 1.45MHz$. The number of required taps according to MATLAB CODE 2 is 32. Figures 22 and 23 provide the form of the I signal, after the CIC filter and, after the FIR filter and second decimation respectively, while Figure 24 presents the transfer function of the FIR filter. Figure 23 demonstrates that in the receiver, prior to decision, the main lobe (0-1.453125 MHz) and, the first side lobe (1.453125-2.90625 MHZ) of the signal, has been received.

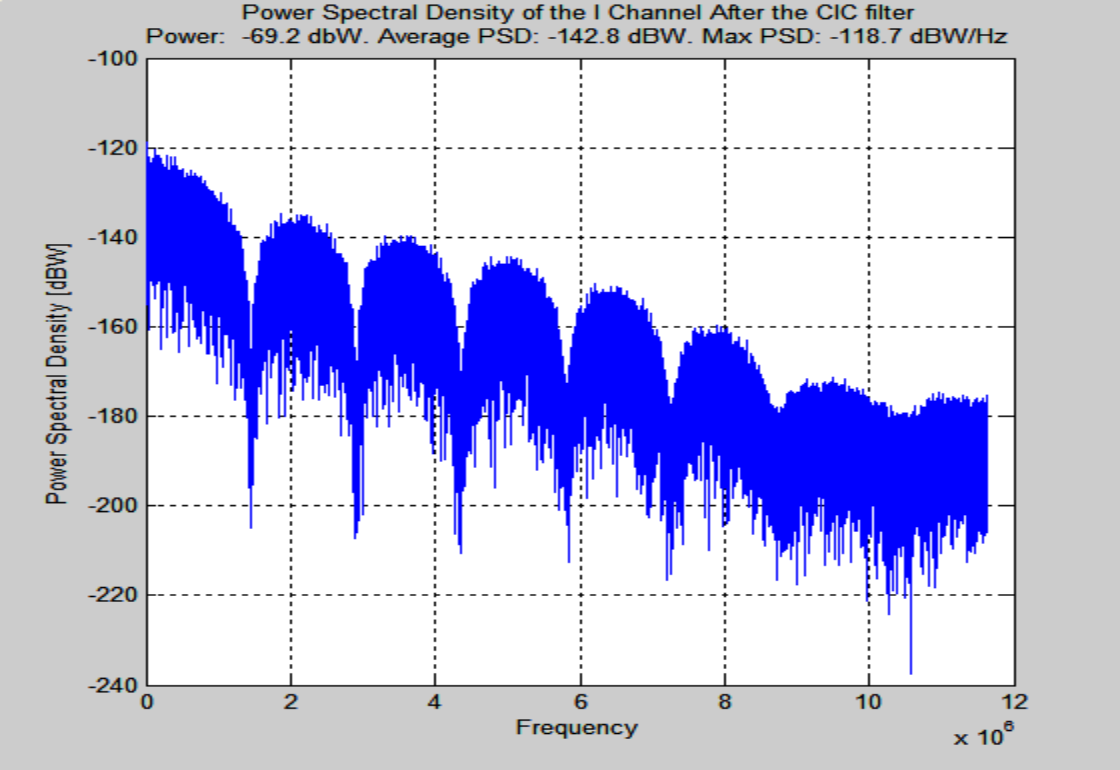

Figure 22. PSD of the Received I Signal after the CIC Filter.

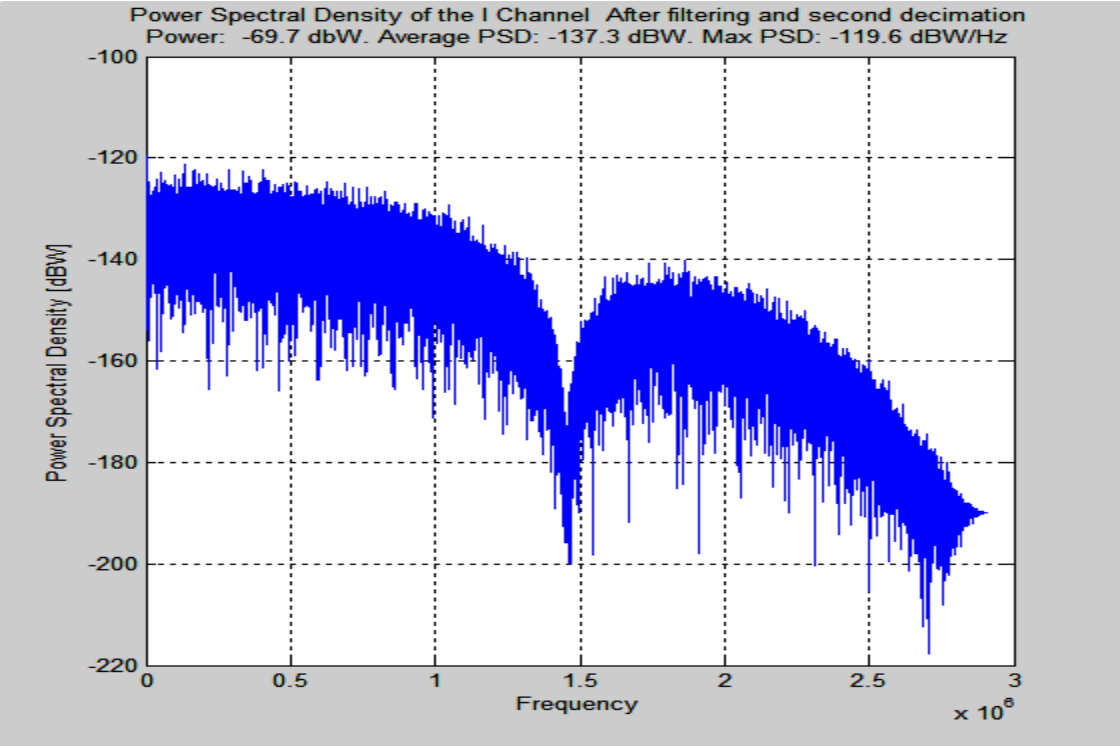

Figure 23. PSD of Received I Signal after Filtering and Second Decimation in FCE.

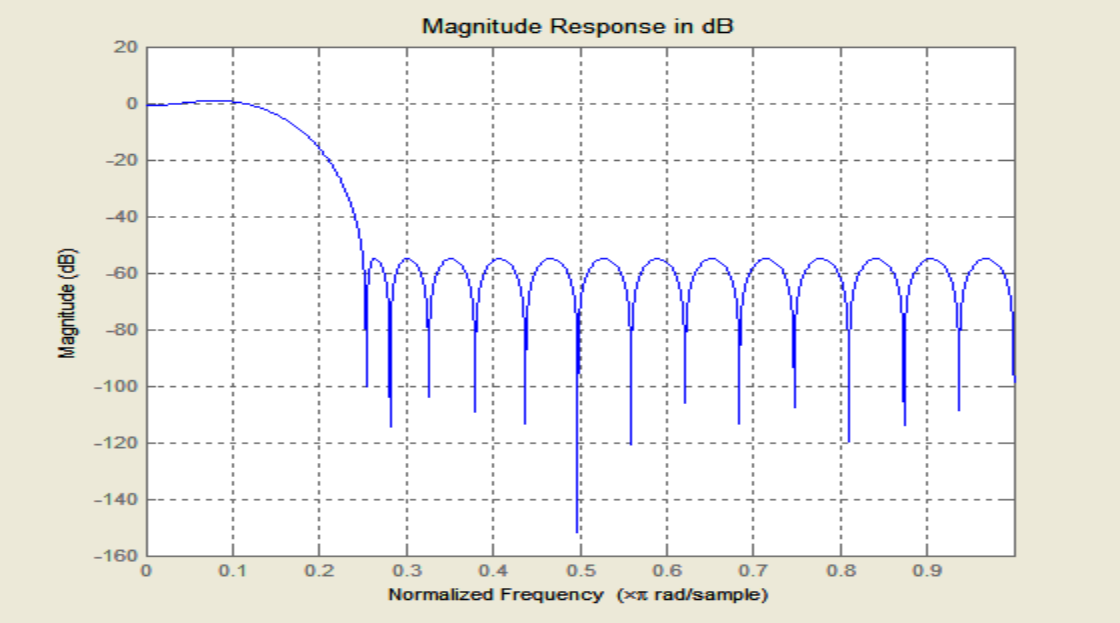

Figure 24. Transfer Function of the FIR Filter Used Prior to the Second Decimation.

- Integration using **a part** of the remaining samples per symbol after the above two decimations. An explanation of why all the remaining samples per symbol are not used appears later.
- Decision at the output of the receiver about the received symbol and extraction of BER vs. the ratio $\frac{Eb}{No}$ where Eb is the average energy per bit and No is the power spectral density of noise.

B. PERFORMANCE OF THE SYSTEM IN THE PRESENCE OF AWGN

This section investigates the performance of the system by applying different scenarios.

Keeping the same R_s (i.e. $R_s = 1.453125 Msps$), the simple FIR filters used in the transmitter after the interpolation and, in the receiver prior to the second decimation, are replaced with square root raised cosine FIR filters[3]. For the new filters, the pass band-

[3] The configuration tool that accompanies the card allows us to build the desired FIR filter in the transmitter (shaping filter) and in the receiver (Filter Compute Engine).

cutoff frequencies and the number of taps are maintained[4]. The simulation demonstrates the results in Figures 25 through 27. When square root raised cosine FIR filters are used instead of simple FIR filters, the ISI in the receiver is reduced, which leads to the improvement of the system's performance. In both cases, the performance of the system is inferior to the theoretical case. The simulations will be continued using square root raised cosine FIR filters.

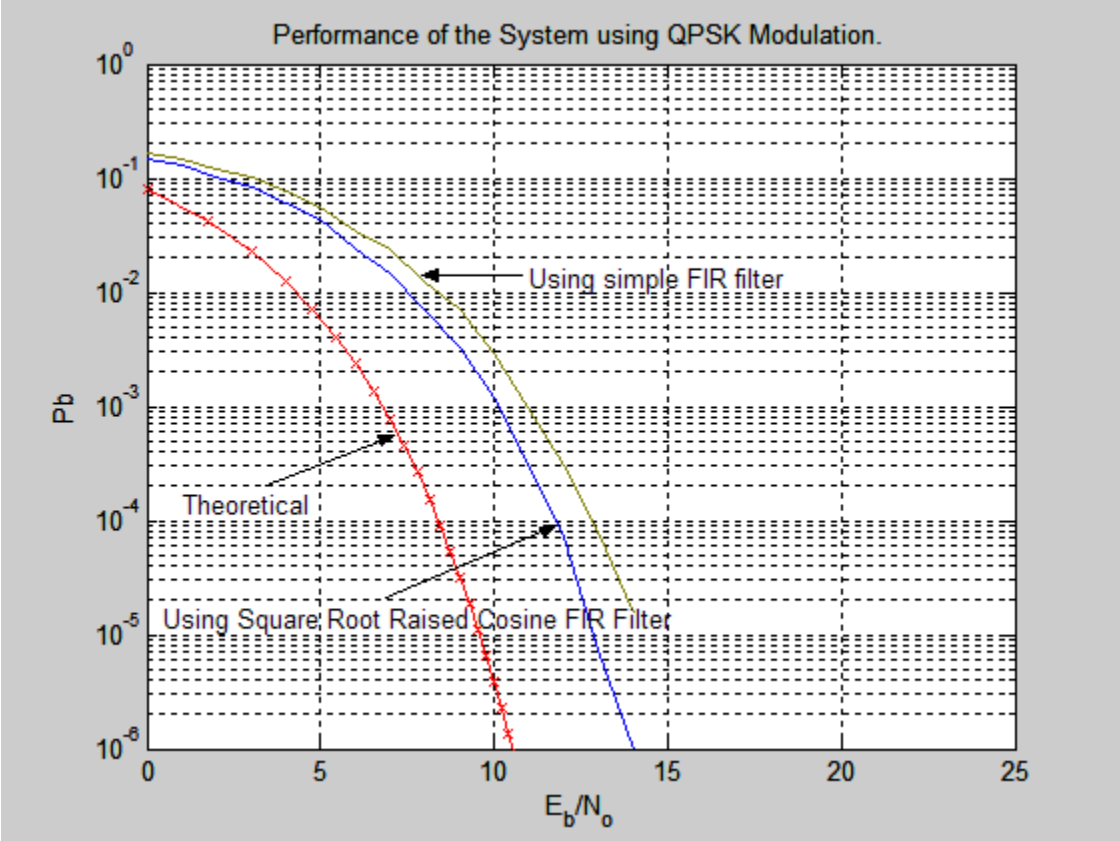

Figure 25. Performance of the System using QPSK Modulation for Different Types of Filters.

[4] For the creation of square root raised cosine FIR filter, the command firrcos(Order of filter, Cutoff frequency, transition region, sampling frequency, 'sqrt') is used where sqrt declares the kind of the filter. From the above syntax it is clear that the command "firrcos" does not alone define the order of the filter, which is user defined. So in the command "firrcos", we use as order of the filter, the same order that is given by the command "remezord", when we use for decimation or interpolation, a simple FIR filter. This order changes each time we use a different interpolation in the transmitter or decimation in the receiver factor. The number of coefficients for different R_{FCE}, R_{CIC} factors, in Table 6, has been derived in this way.

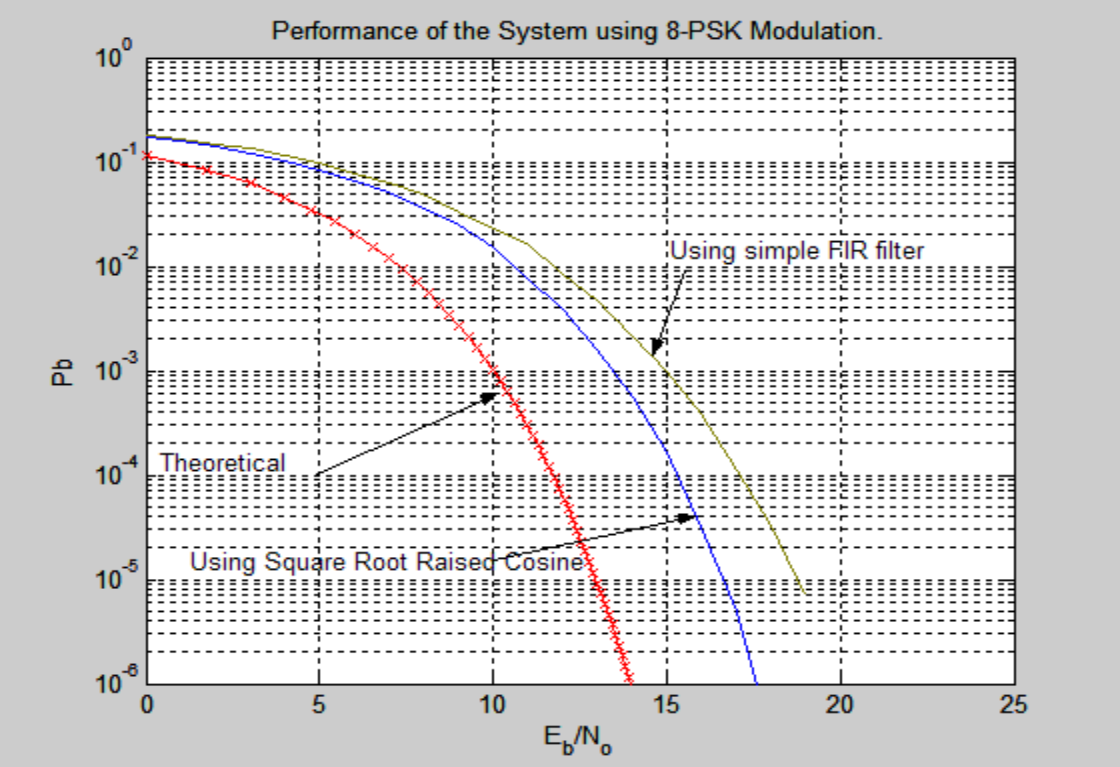

Figure 26. Performance of the System using 8-PSK Modulation for Different Types of Filters.

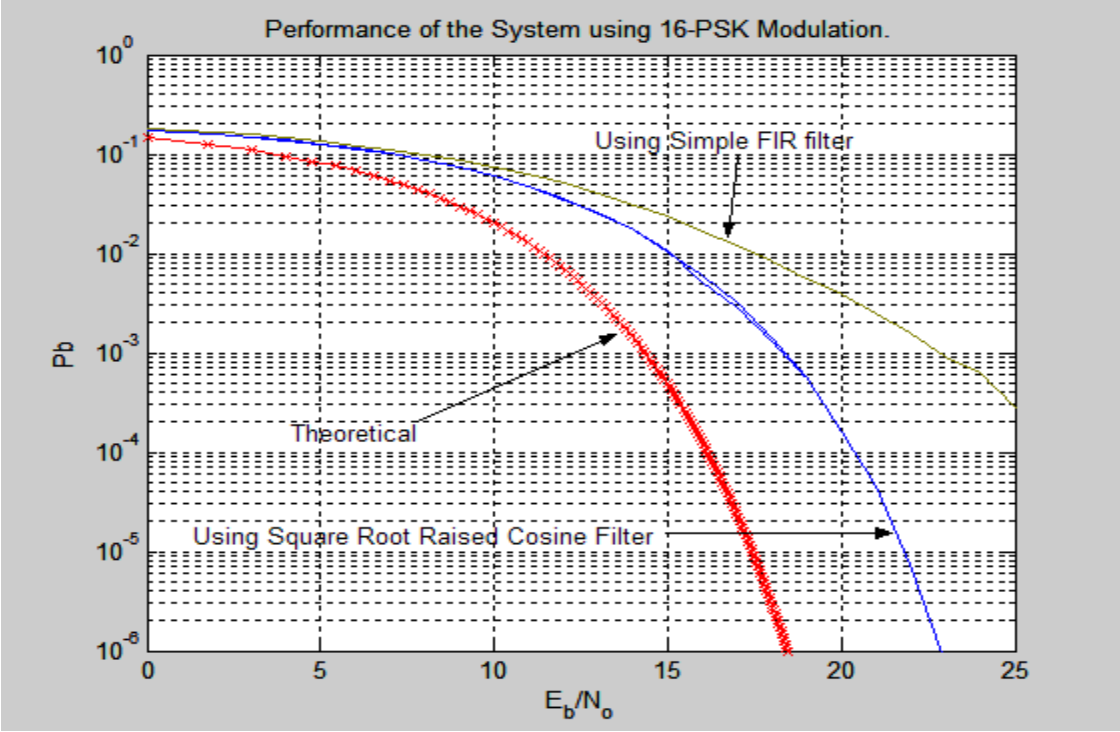

Figure 27. Performance of the System using 16-PSK Modulation for Different Types of Filters.

Thus far, the performance of the system using decimation factors $R_{CIC} = R_{FCE} = 4$, was examined. What happens in the performance of the system if the above values are changed? Based on Figure 28, which provides the functional block diagram of the receiver, the sampling frequency of the baseband signal at the output of the FCE filter is:

$$f_{sfinal} = \frac{f_{sclock}}{R_{CIC} \cdot R_{FCE}} = \frac{93}{(4)\cdot(4)} MHz = 5.8125 MHz . \quad (11)$$

In order to avoid aliasing, the baseband bandwidth of the signal, BW_{BB}[5] according to the Nyquist Theorem, must be

$$BW_{BB} \le \frac{f_{sfinal}}{2} = 2.90625 MHz . \quad (12)$$

In the MPSK modulation this bandwidth expressed in terms of symbol rate R_s, **provided that the main and one side lobe are included**, is given by the equation

$$BW_{BB} = 2 \cdot R_s . \quad (13)$$

It is obvious that the system provides the maximum symbol rate $R_s = 1.453125 Msps$ for $R_{CIC} = R_{FCE} = 4$. Using larger decimation factors, from equations (11), (12), (13), smaller R_s are taken.

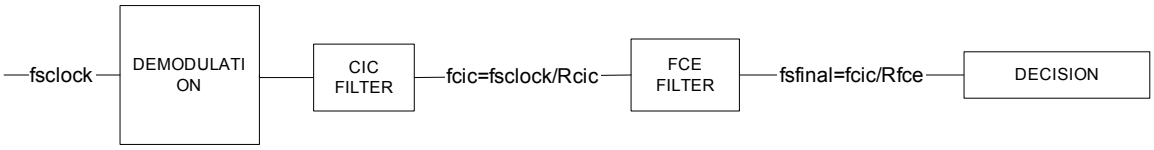

Figure 28. Functional Block Diagram of the Receiver.

After running MATLAB CODE 2 for different values of R_{CIC}, R_{FCE} (Table 6) and, for different symbol rates R_s, imposed by equations (11), (12), (13), the diagram in Figure 29 is obtained. Each time, the program determines the number of coefficients for

[5] The corresponding passband bandwidth is the *bw*.

the FCE filter (see footnote 4) while its cutoff and pass band frequency change, based on the sampling frequency after the CIC filter (i.e. on R_{CIC}) and the decimation factor R_{FCE}. In the first three cases where $R_{CIC} < R_{FCE}$, the performance of the system remains almost the same. In the fourth and fifth case, $R_{CIC} > R_{FCE}$ is used and the performance of the system deteriorates. This is reasonable since the CIC filter is a coarse filter, and using high decimation factors, it cuts higher power frequencies resulting in the degradation of the system's performance. Finally in the last case, $R_{CIC} = 4, R_{FCE} = 16$ is used, with the number of coefficients 32 instead of 128. In this case, the transition region of the FCE filter is smooth compared to the case of 128 coefficients, where the transition region of the filter is quite abrupt. A smooth transition region means that more power of the received signal will pass through the filter. Thus, there is an improvement in the performance of the system. This is confirmed by the position of curve 6 in Figure 29. Of course, this is good in the case where the signal exists alone in the channel, only in the presence of AWGN. When it co-exists with other signals in adjacent spectral areas which use the same combination of CIC, FCE filters, then significant power of one signal will fall into the bandwidth of the others, degrading, in this way, the performance. On the other hand, filters with abrupt transition regions, i.e., with large number of coefficients, demand a rather large number of multiplications per second. The solution to this problem is the multistage sampling rate conversion, which requires less computation and provides more flexibility in the filter design. The FCE in the receiver of the card supports this choice.

CASE	R_{CIC}	R_{FCE}	R_s (in Msps)	**Number of Coefficients for FCE filter (determined by the code, see footnote 4)**
1	4	4	1.453125	32
2	4	8	0.7265625	64
3	4	16	0.36328125	128
4	8	4	0.7265625	32
5	16	4	0.36328125	32
6	4	16	0.36328125	32 (code gives 128 as in third case)

Table 6. Different Values for R_{CIC}, R_{FCE} in order to Investigate the Performance of the System.

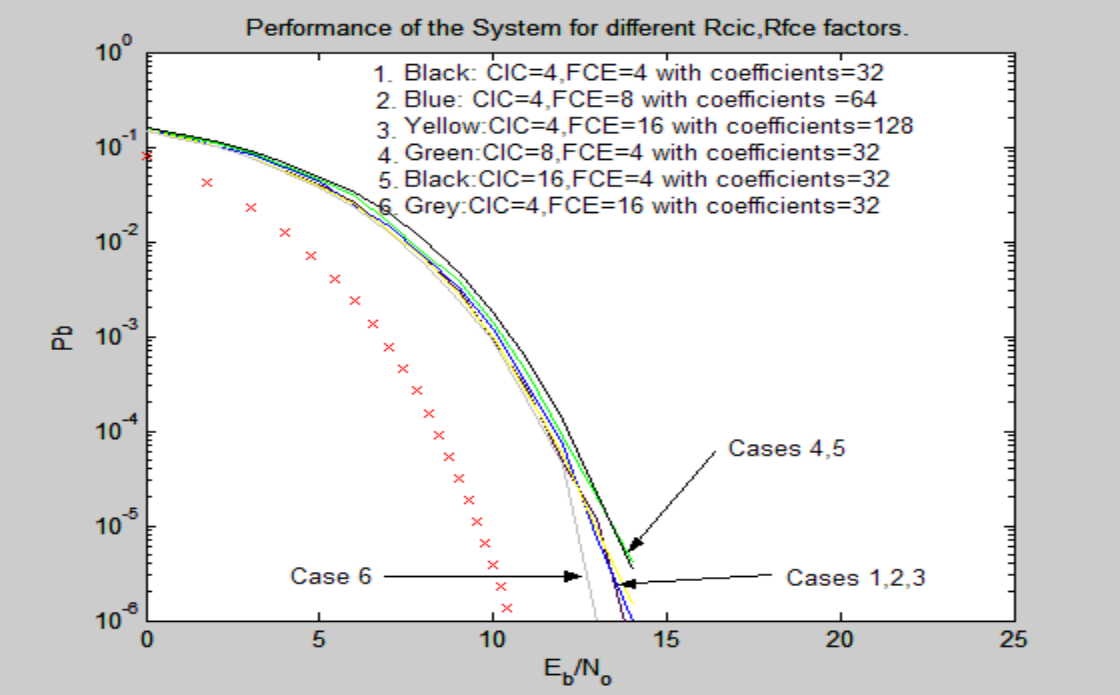

Figure 29. Performance of the System for Different Values of R_{CIC}, R_{FCE} (QPSK Modulation).

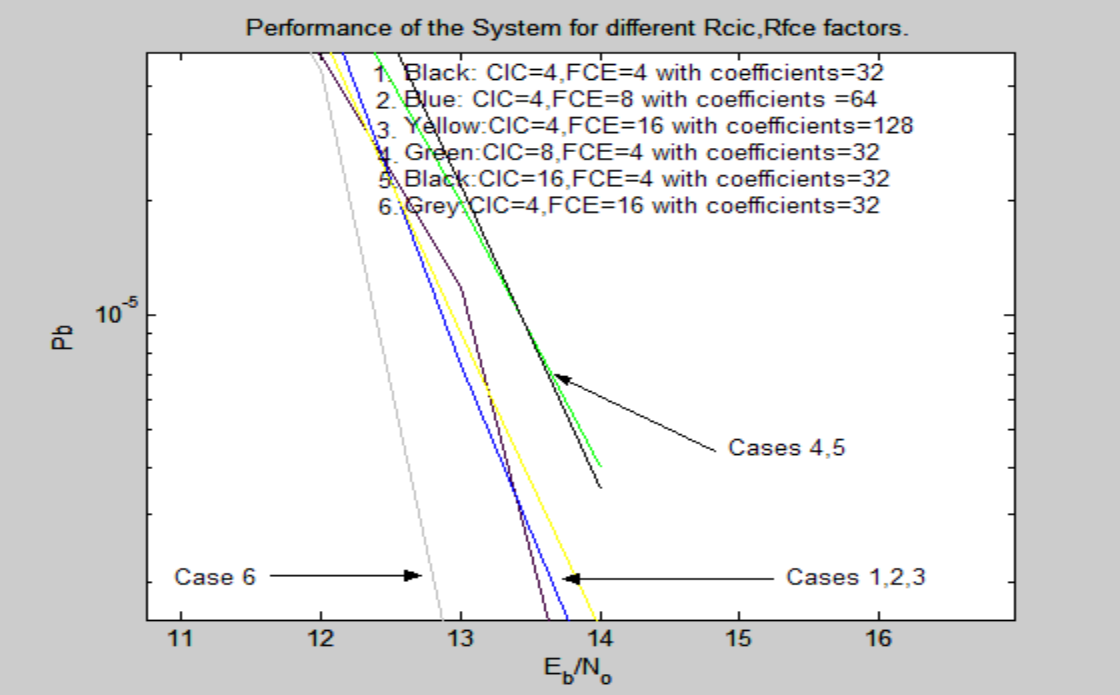

Figure 30. Expanded Lower Region of Figure 29.

The investigation of the system's performance is continued using $R_{CIC} = R_{FEC} = 4$ and the number of coefficients equal to 32. Assume the baseband of the signal contains the main lobe and n side lobes. Then, its baseband bandwidth becomes:

$$BW_{BB} = (n+1) \cdot R_s \leq 2.90625MHz \Rightarrow R_s \leq \frac{2.90625}{n+1} Msps. \quad (14)$$

The signal's rate has been reduced. On the other hand, in this manner, the performance of the system has been improved since most of the signal's energy contained in the n sidelobes, passes through the FCE filter. Of course, this cannot occur forever as $n \to \infty$. After a value of n, the performance of the system converges because the percentage of the power contained in the higher order side lobes is negligible. Figures 31 to 33 provide the performance of the system for a different number of side lobes, included each time in the baseband bandwidth of the signal. For all of the modulation schemes, the letters represent the following cases:

A represents $R_s = 2.90625Msps$. Baseband bandwidth contains only the main lobe of the signal.

B represents $R_s = 2.90625/2Msps$. Baseband bandwidth contains the main lobe and one side lobe of the signal.

C represents $R_s = 2.90625/3Msps$. Baseband bandwidth contains the main lobe and two side lobes of the signal.

D represents $R_s = 2.90625/4Msps$. Baseband bandwidth contains the main lobe and three side lobes of the signal.

E represents $R_s = 2.90625/6Msps$. Baseband bandwidth contains the main lobe and five side lobes of the signal.

F represents $R_s = 2.90625/8Msps$. Baseband bandwidth contains the main lobe and seven side lobes of the signal.

A significant improvement in the performance of the system between cases A and B is demonstrated, but from cases C-F, it is evident that the performance has converged.

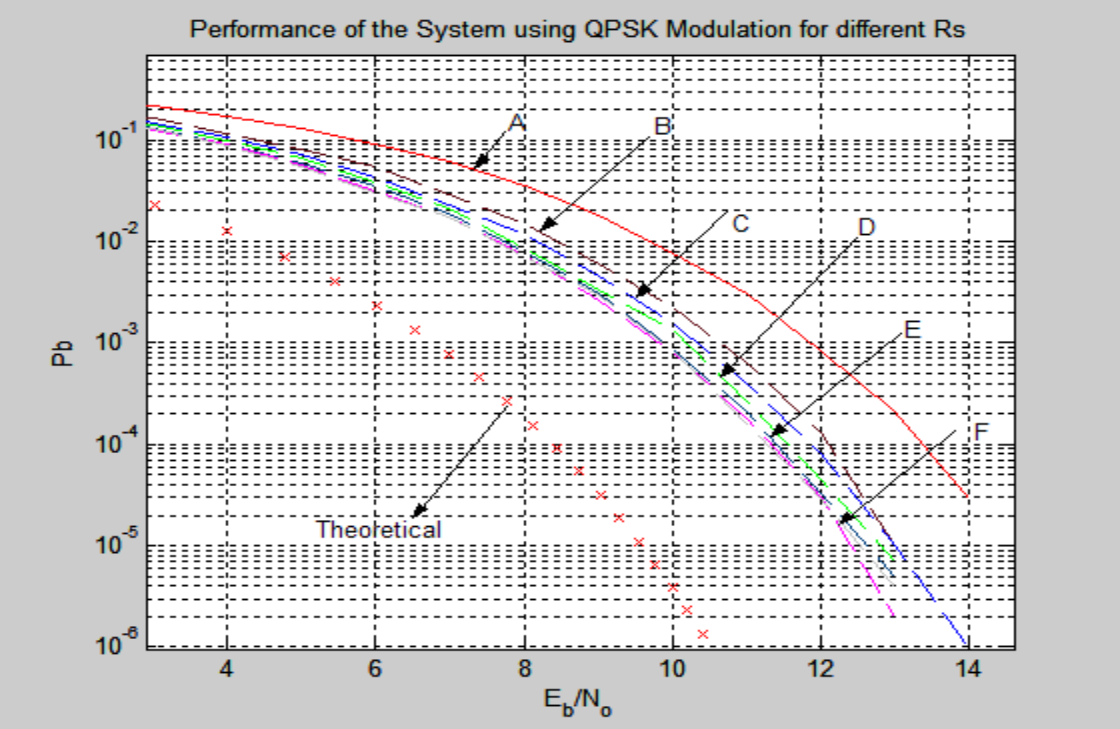

Figure 31. Performance of the System using QPSK Modulation for Different *Rs* .

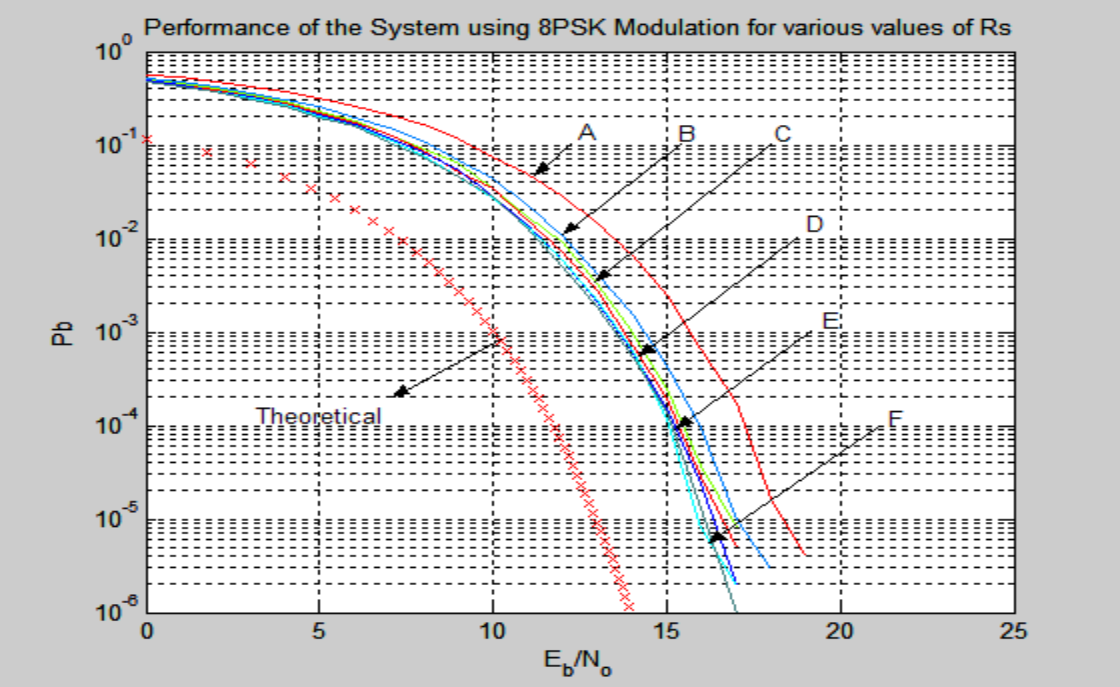

Figure 32. Performance of the System using 8-PSK Modulation for Different *Rs* .

From theory, it is known that in the output of the receiver, in order to make a decision about each separate received symbol, all the values of the samples, referred to a specific symbol, must be added and the number of samples per symbol must divide this sum. In this simulation, the number of samples per symbol after the decimations imposed by the CIC and the square root raised cosine FIR filters, is 4. Using QPSK modulation and with the help of Figure 33, the following conclusion is made. The performance of the system is better upon omitting the first sample from each symbol and using only the last three, in order to decide about the received symbol each time. This happens because upon moving from one symbol to another, a transition region is passed through which always exists between two successive symbols. Thus, the initial samples from each next symbol belong in the transition region and, if these are taken care of in the decision for the received symbol, they act destructively and thus reduce the performance of the system.

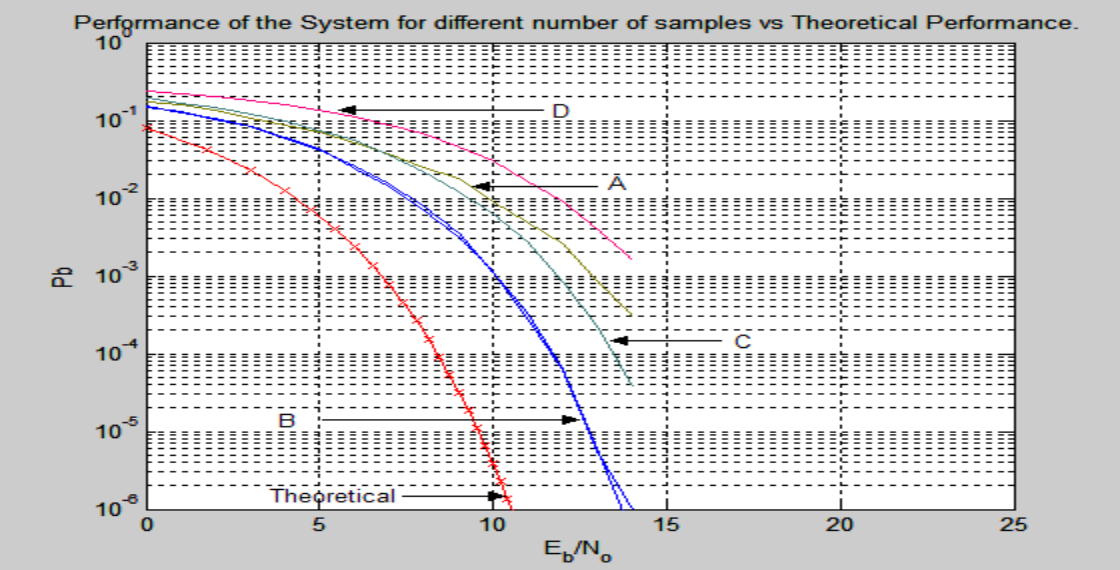

Figure 33. A. Performance of the System using for Decision all the Samples from Each Symbol (samples=4). B. Performance of the System using for Decision the Last 3 Samples from Each Symbol. C. Performance of the System using for Decision the Last 2 Samples from Each Symbol. D. Performance of the System using for Decision the Last One Sample from Each Symbol.

THIS PAGE INTENTIONALLY LEFT BLANK

V. TRANSMISSION OF SIGNALS, COMING FROM QPUC CHANNELS, IN THE LOWER NOISE SPECTRAL AREAS OF THE OPERATING BANDWIDTH OF THE RED RIVER TRANSMITTER CARD, USING A FIXED DATA RATE BY POWER CONTROL

A. THEORETICAL APPROACH

Assume a channel with bandwidth *BW*, equal to the operating bandwidth of the transmitter in the Red-River card (3-40 MHz). This channel is affected by AWGN and interferences, which are undesired signals from other sources. From the "Water Pouring Criterion," in order to communicate effectively through this channel, its n Lower Noise spectral areas (Chapter IV) must be found, and the signals, coming from the dual QPUC channels placed in these areas. This requires, as stated in Chapter IV, that the number n has to range from 1 to 8, for the number of channels in the dual QPUC and, the bandwidth, *bw*, of these areas, must agree with the transmitted bandwidth of the QPUC signals of interest only in the main and the 2 side lobes, i.e., $bw = 4 \cdot R_s$. Each time, the relation below has to hold

$$BW \leq n \cdot bw. \tag{15}$$

Prior to analysis, the following is assumed:

- At any time, the bw for all the lower noise spectral areas in the operating bandwidth of the transmitter is the same.
- The bit error rate Pb for all QPUC channels is the same.
- The modulation scheme used from all QPUC channels is the same MPSK, with k=2, 3, 4.
- The transmitted bit rate from each separate QPUC channel is the same.
- The channel corresponding to the operating bandwidth of the transmitter is stationary.

The objective is, by power control, using the most suitable modulation scheme and the most appropriate number m of the n QPUC channels, $1 \leq m \leq n$, to achieve the maximum m parallel transmission bit rate.

The SNR in the input of the receiver for each channel n, is given by the relation:

$$\frac{S_n}{N_n + J_n} = \frac{Eb_n \cdot Rb}{N_n + J_n} \tag{16}$$

$$S_1 = Eb_1 \cdot Rb$$

$$S_2 = Eb_2 \cdot Rb = Eb_2 \cdot \frac{S_1}{Eb_1} \tag{16a}$$

.....

$$S_n = Eb_n \cdot Rb = Eb_n \cdot \frac{S_1}{Eb_1}.$$

Since it is assumed that the probability of error and the transmitted bit rate for all channels is identical, from equation (16) the equality below must hold

$$\frac{Eb_1}{N + J_1} = \frac{Eb_2}{N + J_2} = ... = \frac{Eb_n}{N + J_n}. \tag{17}$$

From equation (17) the following relations can be extracted.

$$\frac{Eb_2}{Eb_1} = \frac{N + J_2}{N + J_1}$$

$$\frac{Eb_3}{Eb_1} = \frac{N + J_3}{N + J_1} \tag{18}$$

.....

$$\frac{Eb_n}{Eb_1} = \frac{N + J_n}{N + J_1}.$$

N is the noise power, equal for each channel since the noise is AWGN and the bandwidth bw for all the n channels is the same. Replacing equations (18) to (16a), the following is taken:

$$
\begin{aligned}
S_1 &= Eb_1 \cdot Rb \\
S_2 &= \frac{N + J_2}{N + J_1} \cdot S_1 \\
&\ldots\ldots\ldots\ldots\ldots\ldots \\
S_n &= \frac{N + J_n}{N + J_1} \cdot S_1,
\end{aligned}
\tag{19}
$$

with the total received power given by:

$$
S_{totalreceived} = S_1 + S_2 + \ldots\ldots + S_n \,. \tag{20}
$$

The $S_{totalreceived}$ depends on the total transmitted power and the attenuation-fading that each separate channel undergoes due to the changing distance between transmitter and receiver. Assuming that the position of the receiver-transmitter is fixed, no fading occurs and, all the n channels meet the same attenuation, then the total received power is known since:

$$
S_{totalreceived} = S_{totalTransmited} - L \cdot S_{totalTransmited}\,, \tag{21}
$$

where L is the attenuation factor. From equations (19) and (20), the received power for the first channel is found:

$$
S_1 = \frac{S_{totalreceived}}{1 + \dfrac{(n-1) \cdot N + J_2 + \ldots + J_n}{N + J_1}}, \tag{22}
$$

while from the group of equations (19) the same quantity for the other $n-1$ channels is derived. In this way, the total received power, and by extension the total transmitted power, has been allocated among the channels. The common bit rate Rb for all channels is given by the equation:

$$Rb = \frac{0.96 \cdot S_1}{Eb_1} = \frac{\frac{0.96 \cdot S_1}{No}}{\frac{Eb_1}{No}} \quad . \tag{23}$$

In equation (23) the received power S_1 has been multiplied by the factor 0.96 because the main and the two side lobes contain 96% of the total power of the signal.

From the same equation, in order to calculate the bit rate, the ratio Eb_1 / No must be known, except for the received power S_1, since the psd of the thermal noise No, for the system, is known or can be measured. From theory, the probability of bit error rate for a MPSK system [9], in the presence only of AWGN is given by the following equation:

$$Pb = \frac{2}{k} \cdot Q[\sqrt{\frac{2 \cdot k \cdot Eb}{No}} \cdot \sin\frac{\pi}{2^k} \quad . \tag{24}$$

From this equation, the ratio $\frac{Eb}{No}$ can be derived which is required for a specific Pb. Thus:

$$\begin{aligned} & Q(z) = Q[\sqrt{\frac{2 \cdot k \cdot Eb}{No}} \cdot \sin\frac{\pi}{2^k}] \Rightarrow z = \sqrt{\frac{2 \cdot k \cdot Eb}{No}} \cdot \sin\frac{\pi}{2^k} \\ & \Rightarrow \frac{Eb}{No} = \frac{1}{2 \cdot k}(\frac{z}{\sin\frac{\pi}{2^k}})^2 . \end{aligned} \tag{25}$$

Unfortunately in this case, the above equation cannot be used for two reasons. The first reason is the performance of the card that, as seen in Chapter V, in the presence of AWGN, does not coincide with the performance of a theoretical MPSK system. The second reason is that the above equation holds only in AWGN. In our case, except for AWGN, there are random interferences. Thus, in order to calculate the ratio $\frac{Eb}{No}$, it is necessary to proceed with simulation.

B. SIMULATION FOR THE CALCULATION OF $\frac{Eb}{No}$

For this simulation MATLAB CODE 3 is used. It is assumed that the transmitted MPSK signal coming from a QPUC channel is centered at $f_c = 30MHz$. It is between an 8-FSK and a QPSK signal at 20 MHz and 40 MHz respectively, which represent the interferences in the channel. All the above signals have different bit rates, while the QPSK and the MPSK have different initial phases in a trial to achieve the randomness of interference signals. When the two interferences are moved toward the signal, the power of the interferences J that falls into the bandwidth of the transmitted signal--which is equal to $4 \cdot R_s$ since the interest is only in the main and the two side lobes--changes. In this manner, the family of parametric curves is obtained as shown in the figures below, which indicate how the $\frac{Eb}{No}$ changes each time in relation to Pb, for different $\frac{J}{S}$ ratios and modulations. If the ratio J/S in the signal's bandwidth is known, the most appropriate curve for a specific modulation scheme can be chosen and, based on the desired P_b the ratio Eb/No can be derived. Replacing it in equation (23), the bit rate R_b is obtained.

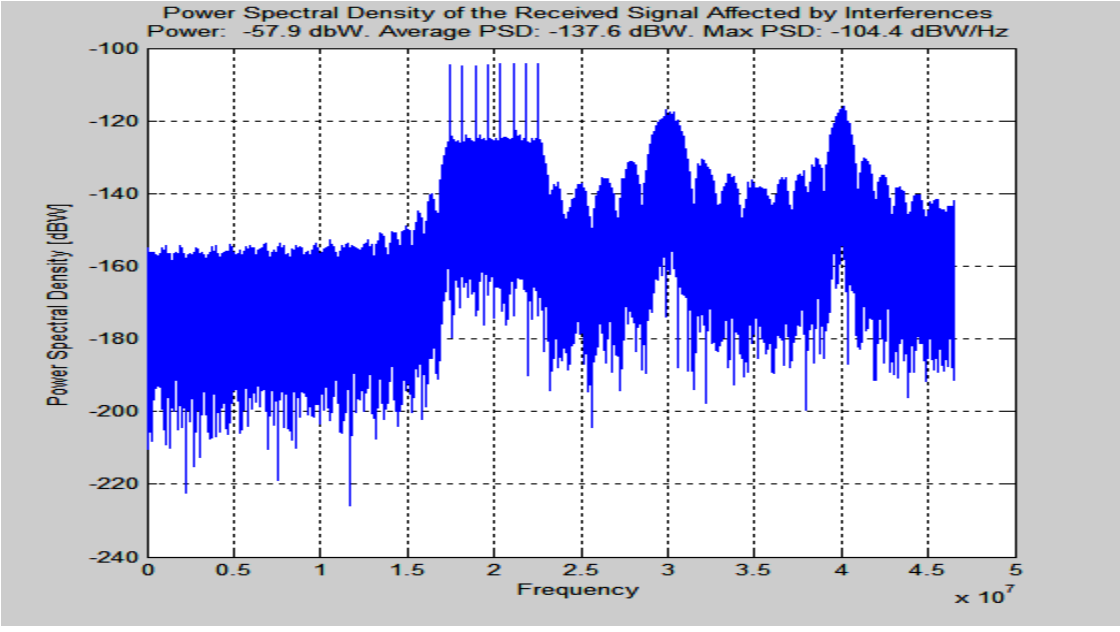

Figure 34. Transmitted Signal Plus Interferences.

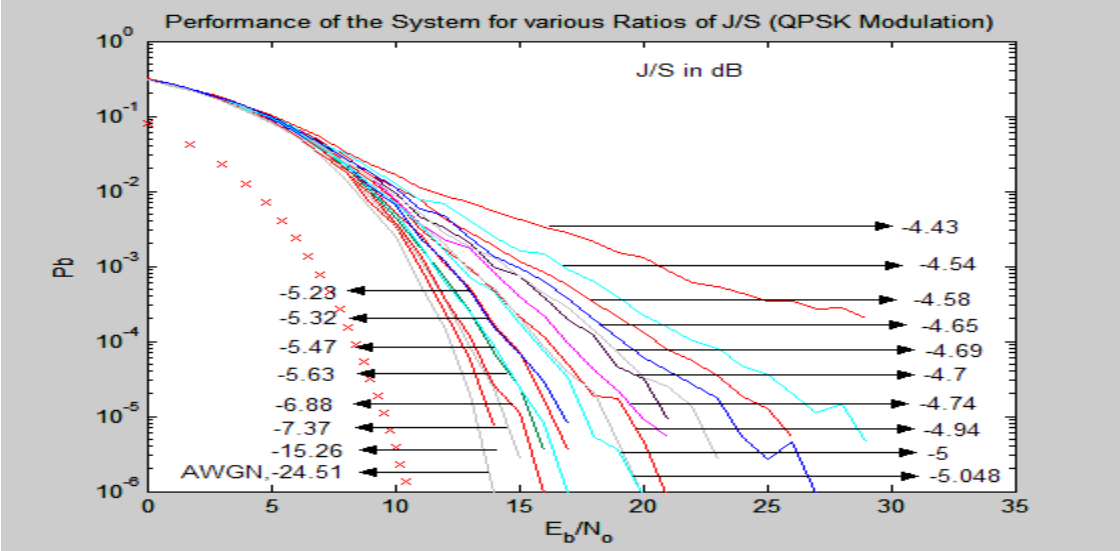

Figure 35. Performance of the System for Various Ratios of J/S (QPSK Modulation).

Carrier of first Interference (in MHz)	**Carrier of second Interference (in MHz)**	**Ratio J/S (in dB)**	**Ratio Eb/No (in dB)**
20	40	-24.5106 Same as in presence only of AWGN	14
24.3	34.7	-15.2631	14.5
25.3	34.7	-7.3719	15
25.5	34.5	-6.8883	15.5
25.9	34.1	-5.6312	16
25.95	34.08	-5.4710	16.5
26	33.9	-5.3239	17
26	34	-5.2368	17.5
26.07	33.95	-5.0048	18
26.1	33.9	-5	19
26.1	33.8	-4.9418	20
26.15	33.85	-4.7478	21
26.2	33.8	-4.7	22
26.2	33.7	-4.7819	23
26.23	33.7	-4.6537	24
26.25	33.7	-4.5895	26
26.3	33.7	-4.5424	29
26.4	33.7	-4.4367	---(converges at high values of Eb/No for Pb=10^-5)

Table 7. Measurements of Ratio J/S and the Corresponding Eb/No for $Pb = 10^{-5}$ for QPSK Modulation.

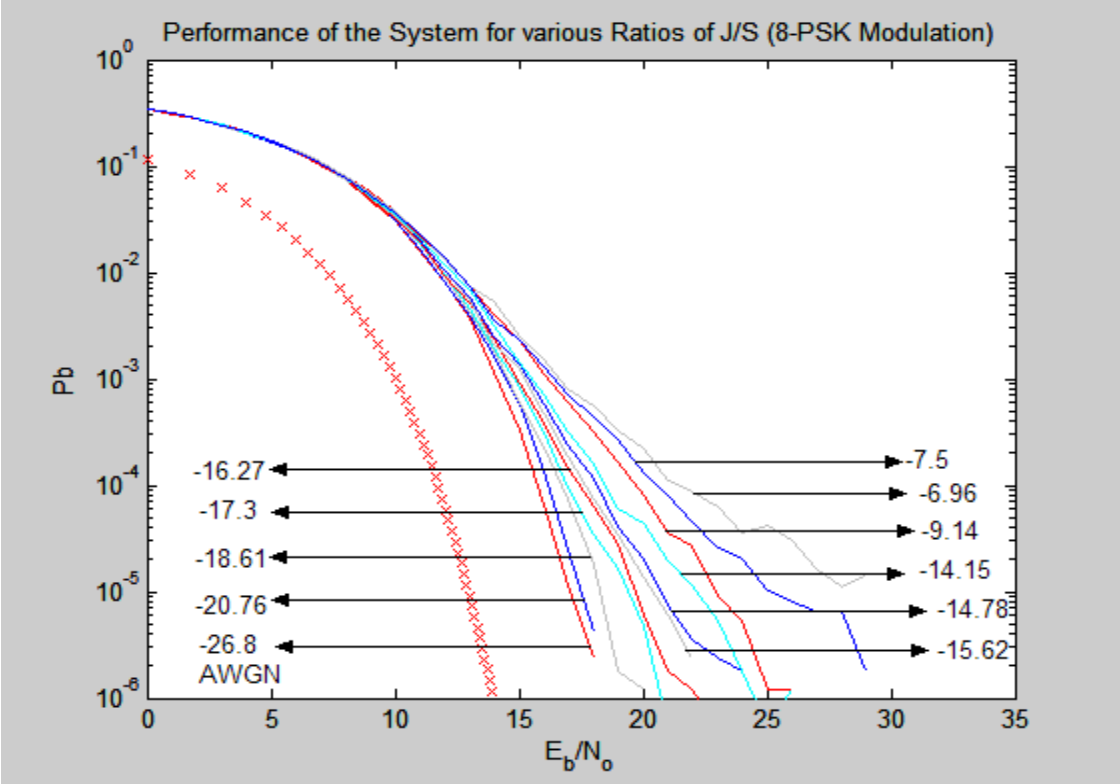

Figure 36. Performance of the System for Various Ratios of J/S (8-PSK Modulation).

Carrier of first Interference (in MHz)	**Carrier of second Interference (in MHz)**	**Ratio J/S (in dB)**	**Ratio Eb/No (in dB)**
20	40	-26.7996 Same as in presence only of AWGN	18
24	36	-20.7624	18.5
24.2	35.8	-18.6181	19
24.3	35.7	-17.3034	20
24.35	35.65	-16.2725	21
24.4	35.6	-15.6284	21.5
24.45	35.55	-14.7878	22
24.45	34.49	-14.1541	23
25	35	-9.1446	24
25	34.5	-7.3452	26
25.5	34.5	-6.9629	29

Table 8. Measurements of Ratio J/S and the Corresponding Eb/No for $Pb = 10^{-5}$ for 8-PSK Modulation.

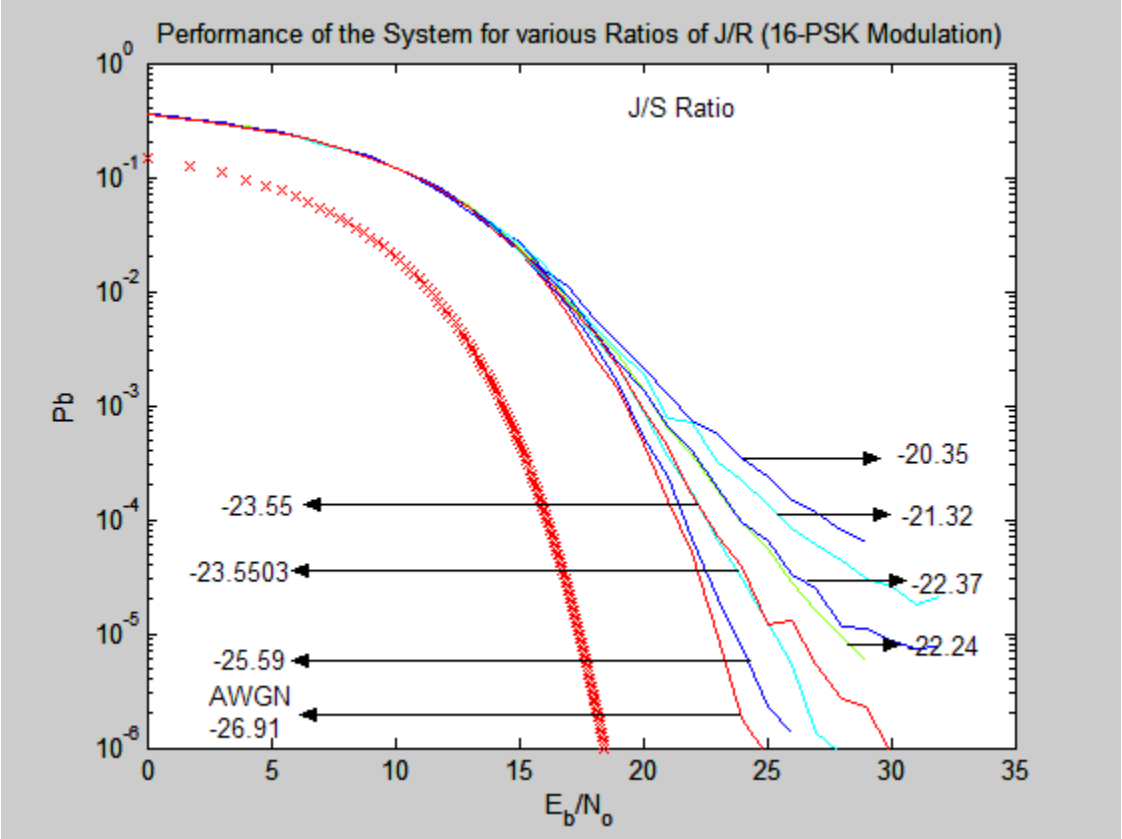

Figure 37. Performance of the System for Various Ratios of J/S (16-PSK Modulation).

Carrier of first Interference (in MHz)	**Carrier of second Interference (in MHz)**	**Ratio J/S (in dB)**	**Ratio Eb/No (in dB)**
23	37	-26.9104	24
23	35.5	-25.5941	24.5
24	36	-23.5503	26
24	35.9	-23.55	27
24.1	35.9	-22.9424	29
24.1	35.85	-22.3752	30
24.15	35.85	-21.3298	---(converges at high values of Eb/No for Pb=10^-5) “ “
24.2	35.8	-20.3538	“

Table 9. Measurements of Ratio J/S and the Corresponding Eb/No for $Pb = 10^{-5}$ for 16-PSK Modulation.

C. OPTIMUM SOLUTION

The basic steps in order to derive the bit rate for n QPUC channels have been described using a specific scheme of modulation. How the initial objective can be achieved has yet to be answered. The steps to provide the optimum solution are described below.

- After having found the n lower noise spectral areas, the search for the optimum solution using QPSK modulation is begun. From the lower noise channel, equation (22) and the set of equations (19), the allocation of the total received power $S_{totalreceived}$ is determined for each one of the n channels. It is obvious that the noisier the channel, the larger the allocated power.
- From the lower noise channel, based on the ratio $J_1/(0.96 \cdot S_1)$, where J_1 and $(0.96 \cdot S_1)$ is the interference and signal power in the bandwidth bw of the channel and, from Figure 35, the appropriate curve J/S is chosen. From this curve and, for a desired BER, the required Eb/No is found. Equation (23) will give the required bit rate Rb, which is identical for all channels.
- The bandwidth of the transmitted QPSK, signal since the interest is in the main and two side lobes, is $B_{QPSK} = 4 \cdot Rb_{QPSK}/2$. At this point, there are two possibilities.
  - $B_{QPSK} \le bw$ or $Rb_{QPSK} \le bw/2$. The solution Rb_{QPSK} is acceptable, since the bandwidth of the signal is less than the bandwidth bw of the channel.
  - $bw < B_{QPSK}$ or $Rb_{QPSK} > bw/2$. This solution is rejected because the bandwidth of the signal is larger than the bandwidth *bw*. The maximum bit rate that the channel can "suffer" is $Rb_{QPSK} = bw/2$. This value is stored for later use and we proceed with the next type of modulation i.e., 8-PSK.
- The 8-PSK modulation using Figure 36, gives a signal with a different bit rate Rb_{8PSK} and bandwidth $B_{8PSK} = \frac{4 \cdot Rb_{8PSK}}{3} \Rightarrow Rb_{8PSK} = 0.75 \cdot B_{8PSK}$. Comparing B_{8PSK} to *bw*, the following possibilities can again be seen:
  - $B_{8PSK} \le bw$ or $Rb_{8PSK} \le 0.75 \cdot bw$. Then $Rb_{8PSK} \ge Rb_{QPSK}$ or $Rb_{8PSK} < Rb_{QPSK}$ where Rb_{QPSK} is the bit rate stored from the QPSK modulation. In each case, the largest bit rate and by extension the modulation scheme from which it is derived must be accepted as the solution.

- $B_{8PSK} > bw$ or $Rb_{8PSK} > 0.75 \cdot bw$. Again, this solution is rejected because the bandwidth of the signal is larger than the bandwidth bw of the transmission channel, and the maximum bit rate that this channel can transmit is $Rb_{8PSK} = 0.75 \cdot bw$. In this case, this value of bit rate is stored for latter use and we proceed with the next type of modulation, 16-PSK applying again the same steps.

- The modulation 16-PSK using Figure 37, gives a signal with bit rate Rb_{16PSK} and bandwidth $B_{16PSK} = Rb_{16PSK}$. Once again, comparing B_{16PSK} to *bw*, the following possibilities can be seen below:
  - $B_{16PSK} \leq bw$ or $Rb_{16PSK} \leq bw$. In this case, Rb_{16PSK} is compared to the bit rates Rb_{QPSK}, Rb_{8PSK} stored from above and the largest is chosen, and by extension, the modulation from which it is derived.
  - $B_{16PSK} > bw$ or $R_{16PSK} > bw$. This solution is rejected and defined as bit rate, $Rb_{16PSK} = 2 \cdot bw$.

The parallel transmission bit rate from the n channels is given by the product of the bit rate, from the above analysis, and the number of channels n:

$$R_{btotal} = n \cdot Rb. \tag{26}$$

The same procedure is repeated and each time the noisiest channel is removed until finally $n = 1$. The best solution is that which provides the maximum parallel transmission bit rate, for a specific number of channels $m, 1 \leq m \leq n$ and, for a specific modulation scheme. When the number of channels m, the bit rate, and the modulation scheme that must be used are set, m signals coming from m QPUC channels are transmitted, following the main functions in the transmitter and in the receiver, of the Red-River Card (Chapter V). The entire procedure for finding the optimum solution is described in the Figure 38.

From time to time, a new problem arises from transmitting the above m signals. A part of the transmitted power of one signal falls into the bandwidth of the others, resulting in the degrading of the performance, i.e., after simulation, there is a bit error rate (BER) which is larger than the predefined acceptable level. The seriousness of this problem, assuming that the $S_{totalreceived}$ is fixed, depends on the position of the m transmitted signals in the operating bandwidth of the transmitter, which in turn, depends on the position of the m lower noise spectral areas, coming from the random noise

profile. There are many approaches to solve this problem. MATLAB CODE 4 shrinks the bandwidth of the lower noise spectral areas, and by extension, the bandwidth of the transmitted signals, coming from QPUC channels, increasing in this manner, their separation. Thus higher order side lobes, i.e., less power, from one transmitted signal, falls into the bandwidth of the others. The penalty for this is a decrease in the maximum symbol rate R_s of the reduced transmission channel. The next section provides a brief description of code 4.

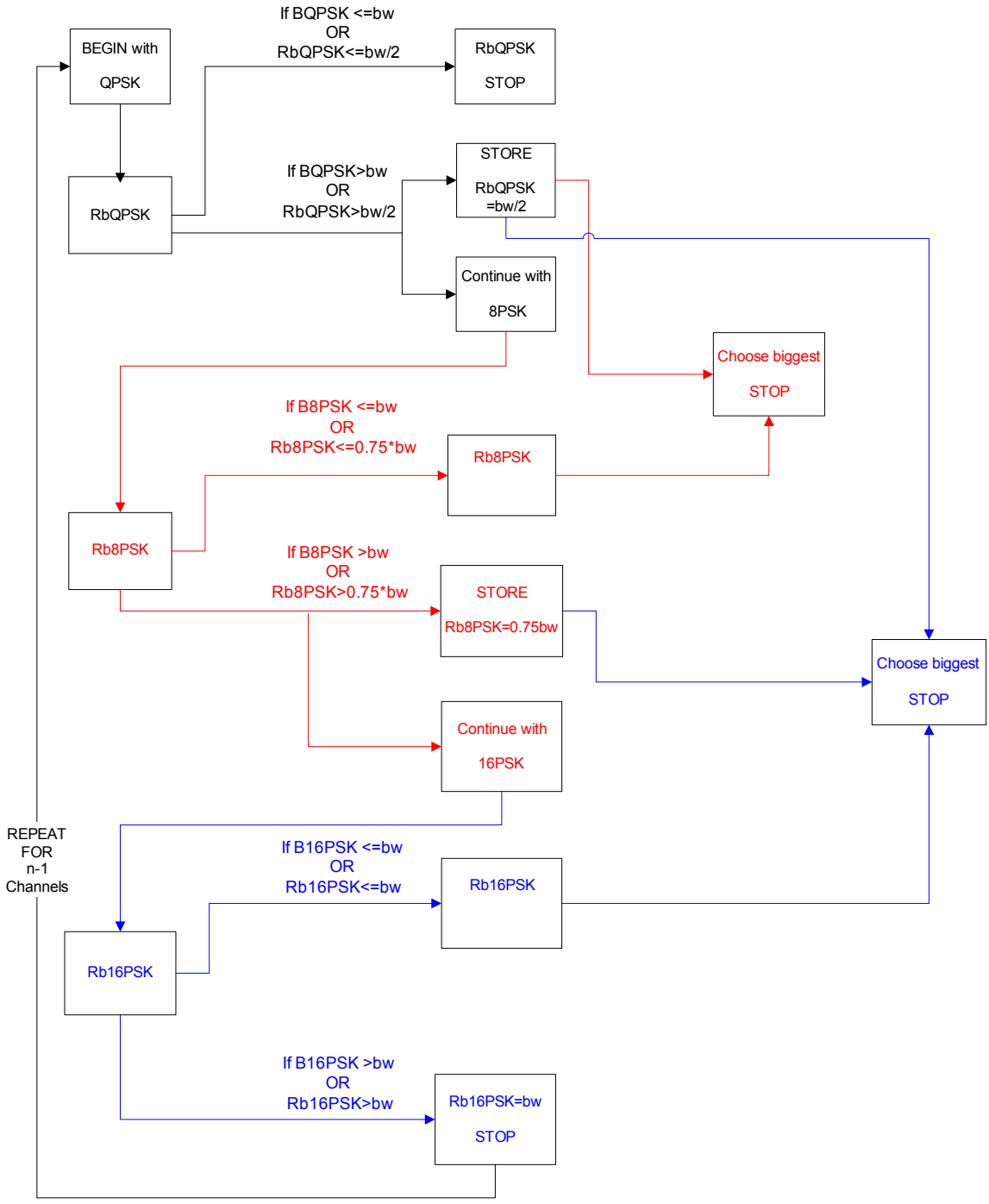

Figure 38. Procedure for Finding the Optimum Solution.

D. BRIEF DEMONSTRATION OF MATLAB CODE 4

Assume that in the operating bandwidth of the transmitter three BPSK interference signals are present at frequencies $fc_1 = 30MHz$, $fc_2 = fc_1/2$, $fc_3 = fc_1/3$ with powers $P1 = 1mW$, $P2 = 1mW$, $P3 = 2mW$ and rates $Rb1 = 3Mbps$, $Rb2 = Rb1/2Mbps$, $Rb3 = Rb2Mbps$, respectively. The power spectral density of the AWGN in the above channel is equal to $No = 10^{-12}$ W/Hz while $S_{totalreceived} = 10^{-3}W$ (Figure 39).

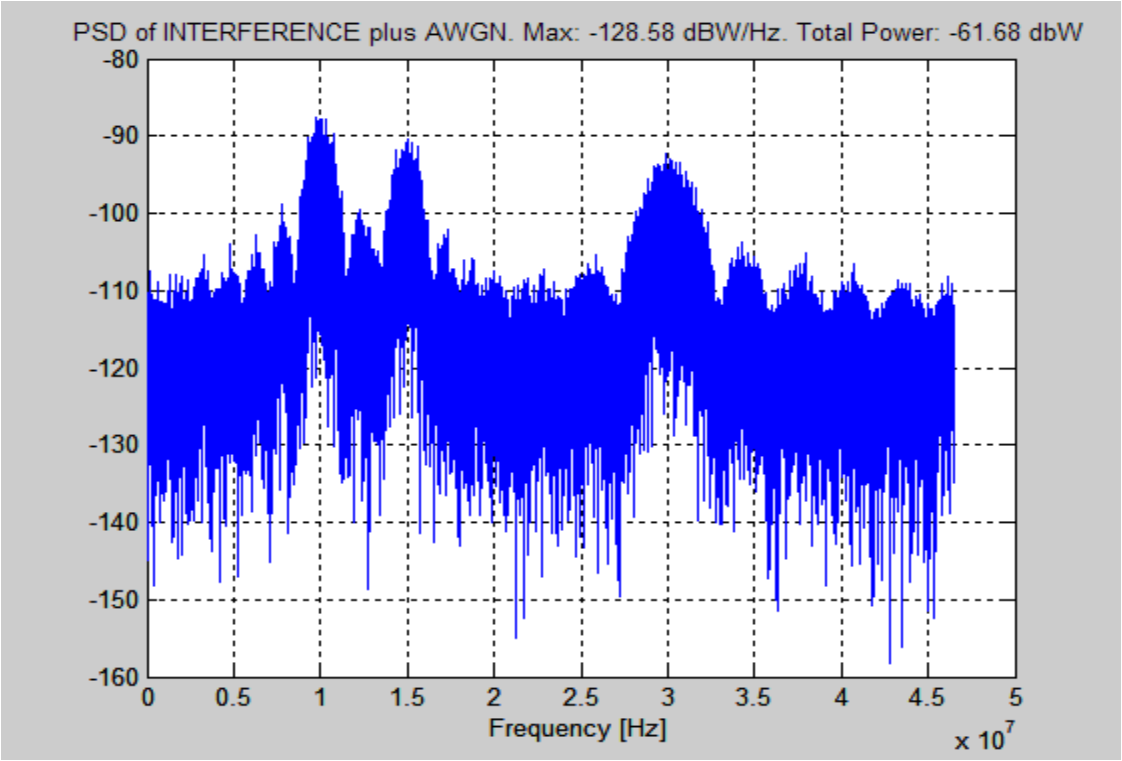

Figure 39. PSD of the AWGN and the Three Interference Signals in the Operating Bandwidth of the Transmitter.

The investigation for the optimum solution begins by trying to find the $n = 8$ lower noise areas in the operating bandwidth of the transmitter, each one with bandwidth $bw = 2.90625MHz$. This bandwidth results from equation (11) for $R_{CIC} = 4, R_{FCE} = 8$ [6].

[6] $R_{FCE} = 4, 5, 6, 7$ can be used, but in this way for values 4,5 equation 15 does not hold. On the other hand, the values 6,7 give a useless transmission bandwidth because a quite significant part of the operating bandwidth of the transmitter is occupied by interference signals.

To be more specific, since the decimation factors of the CIC and FCE filters have the above values, the received signal coming from a QPUC channel will have sampling rate, at the output of FCE filter:

$$f_{sfinal} = \frac{f_{sclock}}{R_{cic} \cdot R_{FCE}} = \frac{93}{4 \cdot 8} MHz = 2.90625 MHz \qquad (27)$$

and baseband bandwidth:

$$BW_{BB} \leq \frac{f_{sfinal}}{2} = 1.453125 MHz \qquad (28)$$

Its transmitted bandwidth will be:

$$BW \leq 2 \cdot BW_{BB} = 4 \cdot R_s = 2.90625 MHz \qquad (29)$$

In order for this signal to be transmittable (main and two side lobes) in the operating bandwidth of the transmitter, a spectral area must be reserved which will have at least a bandwidth of $bw = 2.90625 MHz$.

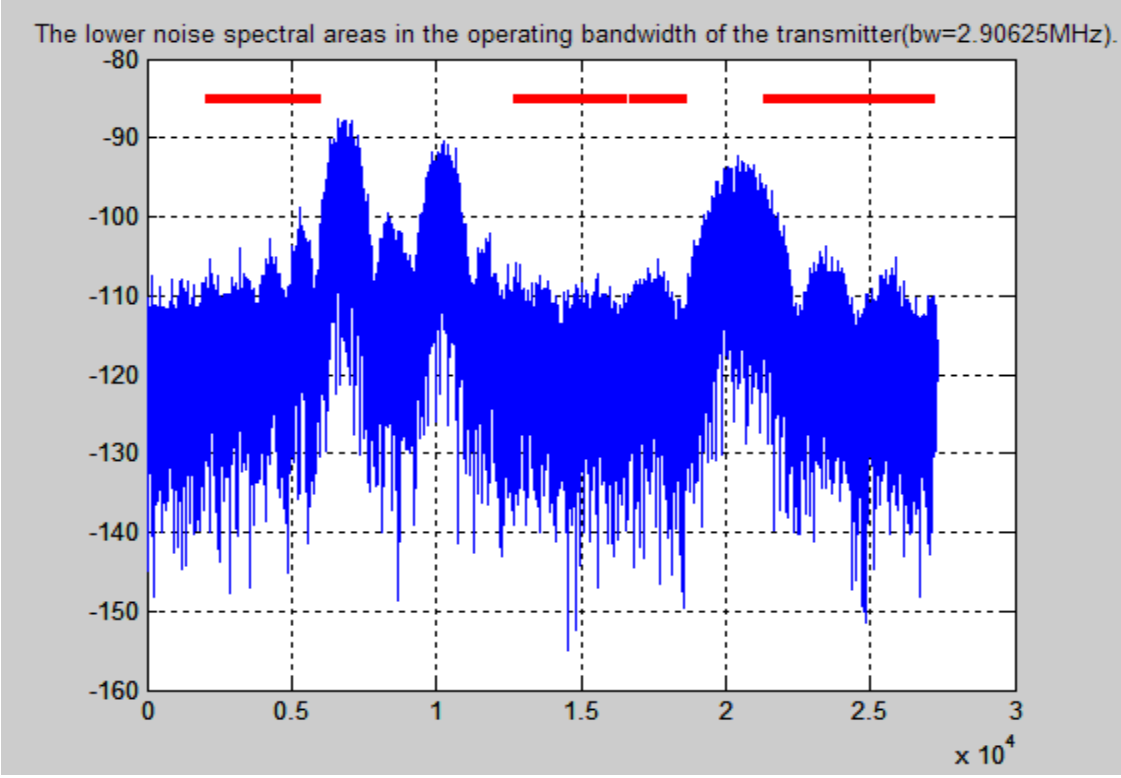

Figure 40. The Lower Noise Areas in the Operating Bandwidth of the Transmitter (bw=2.90625MHz).

The code executes the procedure described in Figure 38 and derives the optimum solution as in Table 10 (6 channels, modulation scheme 8-PSK and, bit rate $R_b = 2.1797 Mbps$). This solution gives the maximum total parallel bit rate, i.e. $6 \cdot (2.1797) Mbps = 13.078 Mbps$. At this point, the code can be checked. In the case of QPSK modulation, the bit rate does not exceed the half of bw. Thus, $Rb_{QPSK} = bw/2 = 2.90625/2 Mbps = 1.4531 Mbps$ is taken. It is obvious that the code has proposed a bit rate larger than $bw/2$, and this justifies the existence of the proposed bit rate for 8-PSK. If the code had proposed a bit rate for QPSK modulation less than $bw/2$, then the bit rate for 8-PSK would be zero. The same happens between the 8-PSK and 16-PSK modulations.

Number of lower noise areas that will be used (n)	6
Bit Rate for QPSK Modulation	1.4531Mbps
Bit Rate for 8-PSK Modulation	2.1797Mbps
Bit Rate for 16-PSK Modulation	0.6339Mbps
Total Parallel Bit Rate (n. Maximum Bit Rate)	13.078Mbps

Table 10. Variable "BEST RESULT". (Gives the optimum solution for bw=2.90625MHz.)

After finding the optimum solution, six 8-PSK signals coming from six QPUC channels are transmitted using the 6 lowest noise spectral areas (Figure 41) in the operating bandwidth of the transmitter, executing the main functions in the transceiver of the Red River Card, as described in Chapter V. Figure 41 demonstrates that the allocation of the total power is different for each signal. In the signals that occupy lower noise areas have allocated less power. In the receiver, at the stage of decision comparing the transmitted to the received symbols, the bit error rate is derived. In our case it is $Pb = 10^{-3}$, i.e., above the predefined level, which is $Pb = 10^{-5}$. Although the optimum solution based on probability of error $Pb = 10^{-5}$ has been found, the simulation gives different results because the 6 transmitted signals are close to one another and, a

significant amount of power of one falls into the bandwidth of the others, acting as strong interference. As stated above, a solution to this problem is to reduce the bandwidth of the lower noise areas, which is identical to the reduction of the bandwidth of transmitted signals and, by extension, to a reduction of transmitted bit rate from each QPUC channel. This shrinking can be achieved by increasing the decimation factor in the FCE filter. In the Red River card, the minimum increase can be equal to one. Due to limitations[7] imposed by Matlab, a minimum increase of 2 is used.

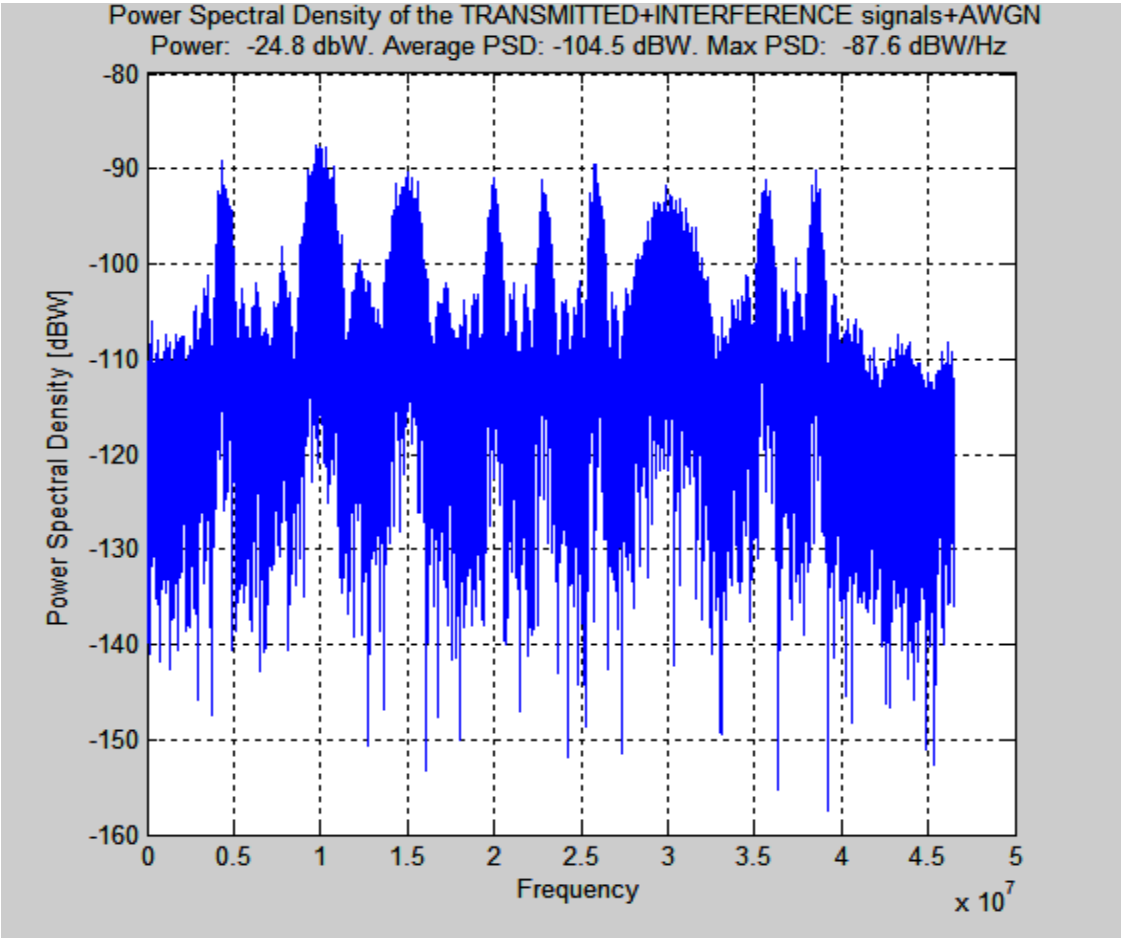

Figure 41. The Six 8-PSK Transmitted Signals in the Presence of Three Interferences and AWGN.

[7] In order to execute the procedures of interpolation, decimation and, integration in Matlab, the command "reshape" is applied. This command requires the division between the number of elements of the matrix and the one desired new dimension of the matrix to be an integer. If it is not an integer, it is necessary to add to or, to remove some elements from the matrix. Thus, errors are intentionally introduced. If the program executes, e.g. 100 loops in order to achieve the transmission of one million symbols, then in each loop, errors are introduced, resulting in a quite high bit error rate.

The code automatically increases the R_{FCE} from 8 to 10. The bandwidth of the lower noise areas becomes $bw = 2.325MHz$ (Figure 42). Now the optimum solution has changed, since the bandwidth and the noise power of the lower noise spectral areas have changed. The optimum solution is given in Table 11.

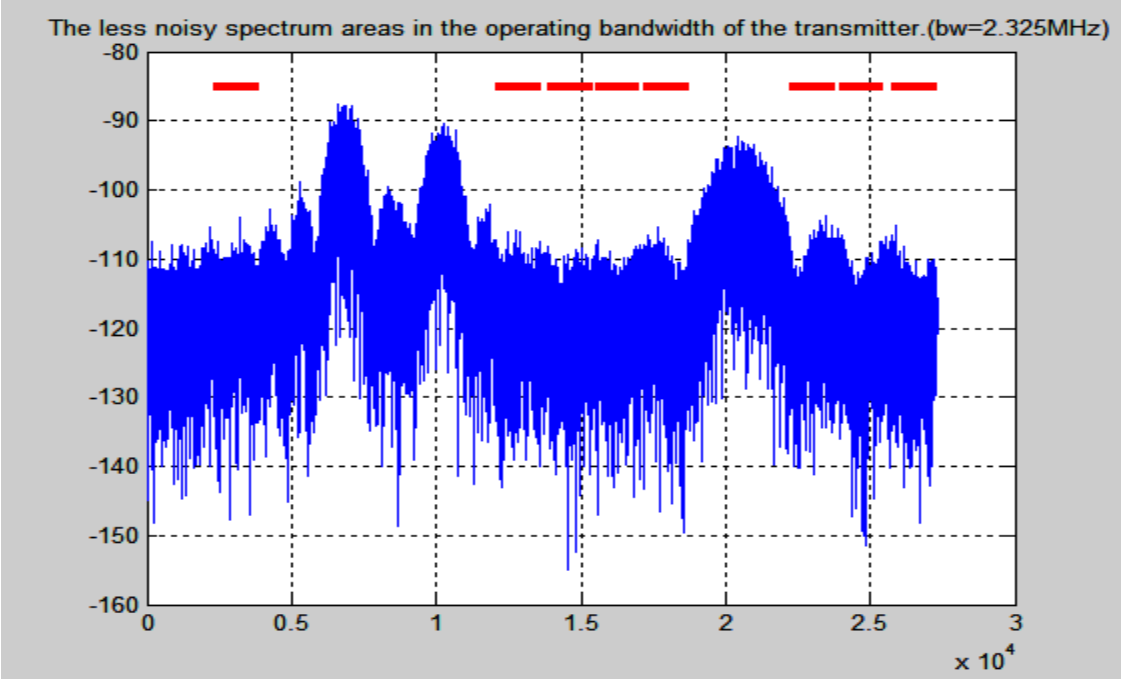

Figure 42. The Lower Noise Areas in the Operating Bandwidth of the Transmitter (bw=2.325MHz).

Number of lower noise areas that will be used (n)	8
Bit Rate for QPSK Modulation	1.1625Mbps
Bit Rate for 8-PSK Modulation	1.5714Mbps
Bit Rate for 16-PSK Modulation	0Mbps [8]
Total Parallel Bit Rate ($n \cdot$Maximum Bit Rate)	12.571Mbps

Table 11. Variable "BEST RESULT". (Gives the optimum solution for bw=2.325MHz.)

[8] It is equal to zero because the proposed by the code bit rate, for 8-PSK, is less than $0.75\,bw$=1.74375Mbps.

Once again, after simulation $Pb = 10^{-4}$. In short, the decimation factor of FCE must be increased to 14 in order for the simulation to give $Pb = 10^{-6}$, which is an acceptable BER since it is less than 10^{-5}. The Figures 43 and 44 and Table 12 provide the accepted optimum solution.

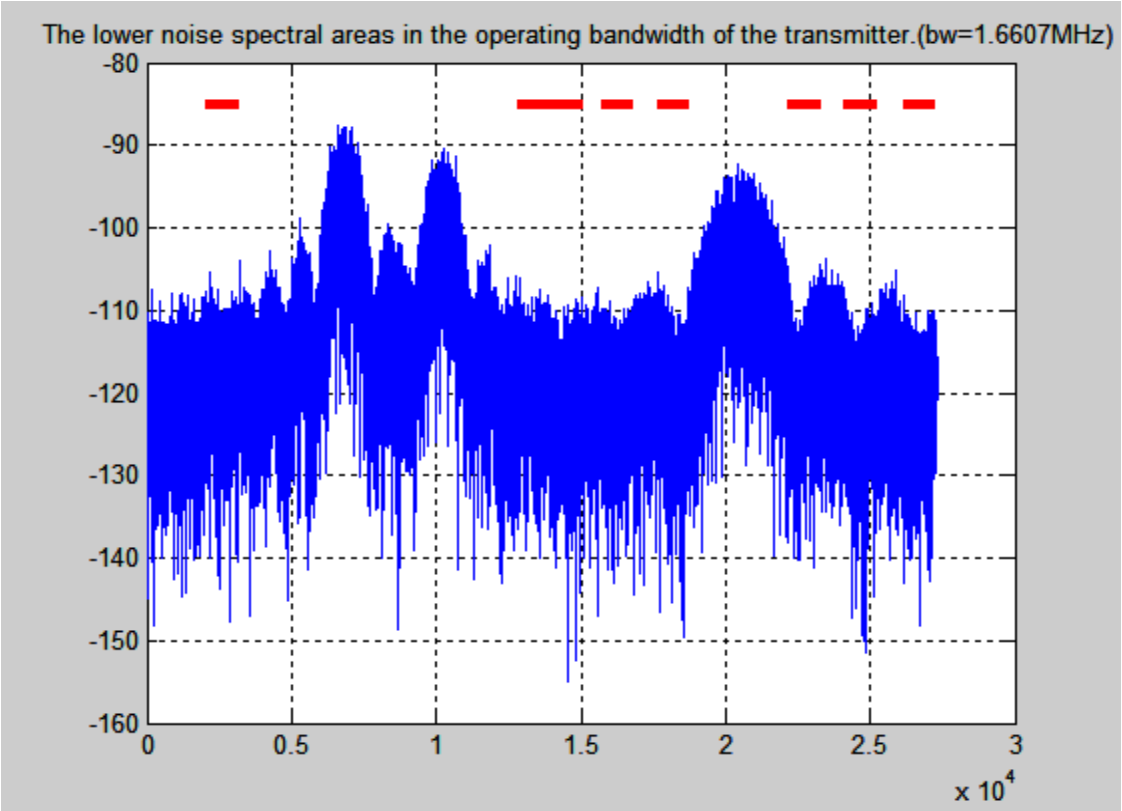

Figure 43. The Lower Noise Areas in the Operating Bandwidth of the Transmitter (bw=1.6607MHz).

Number of lower noise areas that will be used (n)	8
Bit Rate for QPSK Modulation	0.83036Mbps
Bit Rate for 8-PSK Modulation	1.2455Mbps
Bit Rate for 16-PSK Modulation	0.40663Mbps
Total Parallel Bit Rate ($n \cdot$ Maximum Bit Rate)	9.9643Mbps

Table 12. Variable "BEST RESULT". (Gives the optimum solution for bw=1.6607 MHz.)

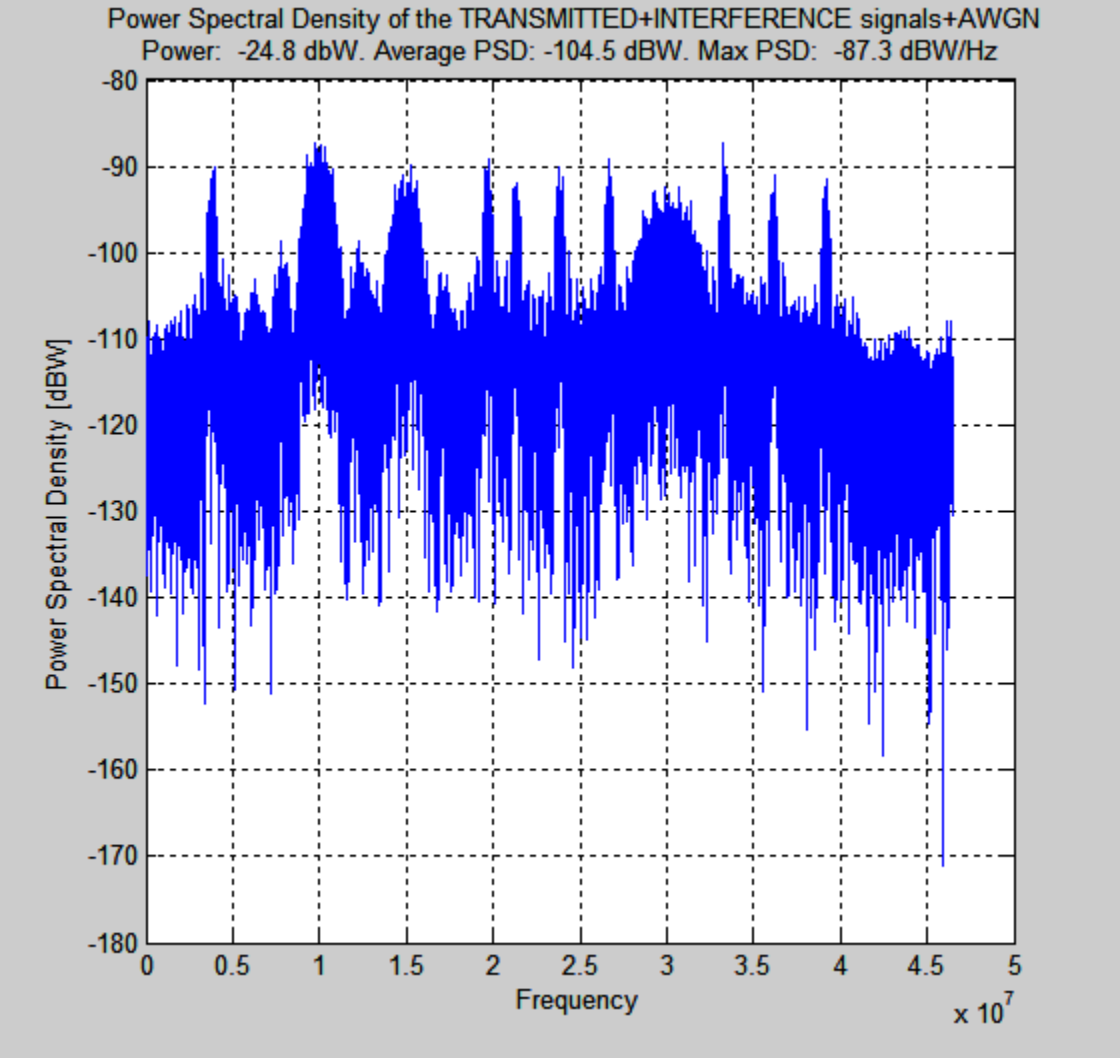

Figure 44. The Eight 8-PSK Transmitted Signals in the Presence of Three Interferences and AWGN.

Finally, the case where the $S_{totalrecei\ ved}$ is quite small is examined. Assume that $S_{totalrecei\ ved} = 10^{-5} W$. In this case, the problem of one signal interfering with the others does not occur. The execution of the code directly gives the optimum solution with BER in the acceptable predefined levels (Figure 45).

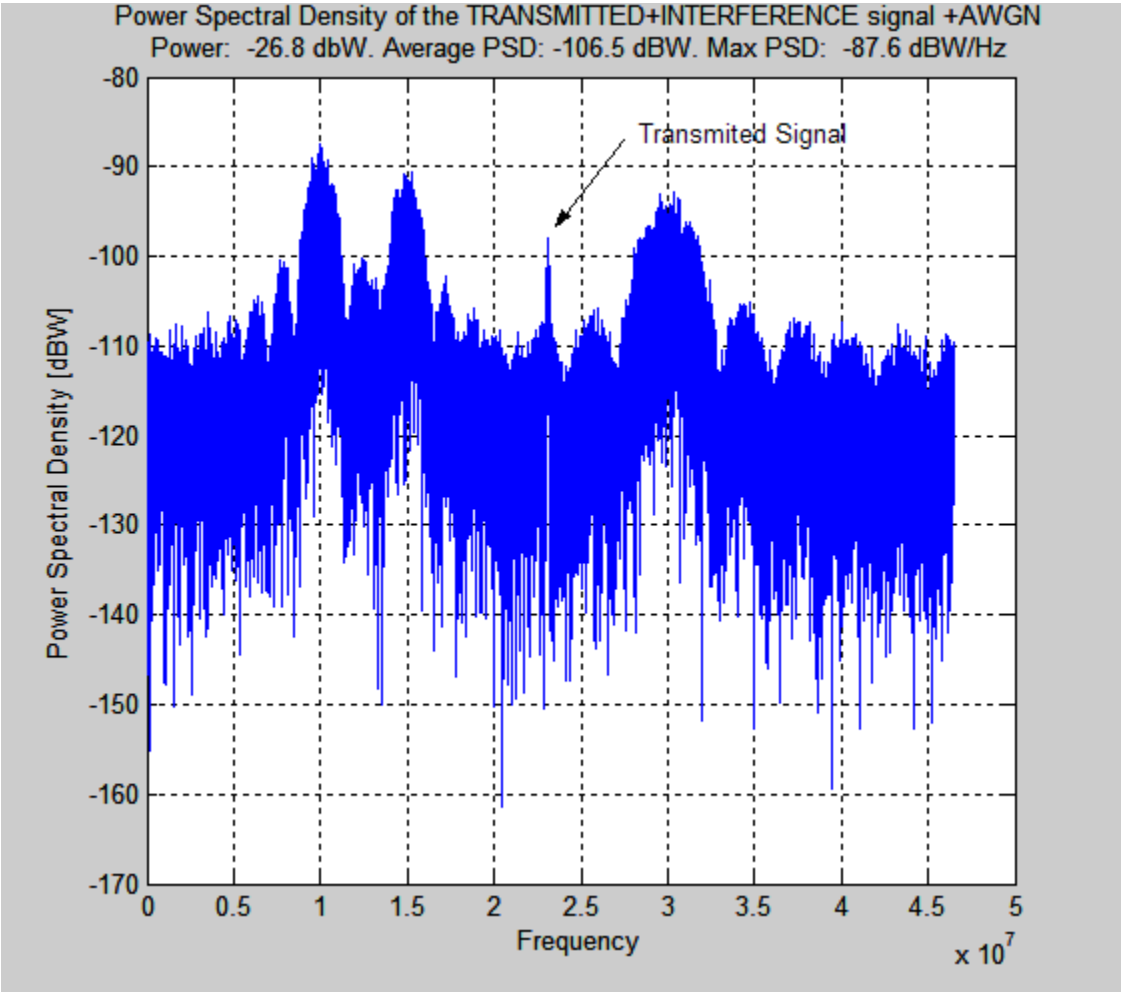

Figure 45. The one QPSK Transmitted Signal in the Presence of Three Interferences and AWGN. Bit rate 0.38218Mbps, $S_{totalrecei\,ved} = 10^{-5} W$, bw=2.90625MHz.

VI. CONCLUSIONS

A Software Defined Radio (SDR) is a collection of hardware and software technologies that enable reconfigurable system architectures for wireless networks and user terminals. A SDR provides an efficient and comparatively inexpensive solution to the problem of building multimode, multiband, multifunctional wireless devices that can be enhanced using software upgrades. The SDR equipment can be dynamically programmed in software to reconfigure the characteristics of equipment. The same piece of hardware can be modified to perform different functions at different times. This allows manufacturers to concentrate development efforts on a common hardware platform.

The objective of this thesis was, using simulation, to synthesize signals with a dynamically changing power spectral density, in a SDR transmitter, utilizing the most appropriate channels, modulation schemes and transmission rates for the communication, based on the noise profile (AWGN plus interferences) of the link, in order to achieve a performance within some predefined acceptable levels. Chapters IV and V presented the results and analysis. In the simulation, the SDR transceiver used the basic functions of the RED RIVER COMPANY card, model 253 WaveRunner Plus PCI, which is a polychannel programmable digital transceiver possessing a high performance SDR capability.

The method to derive the lower noise spectral areas in the operating bandwidth of the above transceiver was demonstrated. Assuming a fixed transmitted power and stationary communication channel, identical to the operating bandwidth of the transmitter, using the high degree of flexibility and precision through the use of programmable digital filters and sample rates, MPSK signals were synthesized. The synthesis of these signals took place, such as control of the fixed transmitted power, using the most appropriate modulation scheme (MPSK with k=2,3,4), the most appropriate number of channels m in QPUC ($1 \le m \le 8$) and the m less noisy spectral areas in the operating bandwidth of the transmitter, to achieve the maximum m parallel transmission

bit rate to keep the probability of error under a predefined acceptable level. In other words, the method of "water pouring" in the frequency domain was applied in order to increase the capacity of the noisy communication channel.

The results clearly indicate that each time the noise profile in the operating bandwidth of the transmitter changes, the parameters of the signals' synthesis (modulation, symbols rate, transmission power, carrier frequency) change dramatically with the objective of increasing the capacity of the communication channel.

Finally, it is shown that after having synthesized the transmitted signals, based on the aforementioned description, the simulation results may have a probability of error larger than the acceptable predefined level. This occurs because the initial noise profile changes after the transmission of the desired signals since each one is interference for the others. The solution to this problem is to reduce the bandwidth of the lower noise areas, which is identical to the reduction of the bandwidth of transmitted signals and, by extension, to the reduction of the transmitted bit rate from each QPUC channel.

APPENDIX

A. MATLAB_CODE_1

%FINDING THE LOWER NOISE SPECTRAL AREAS OF THE
%OPERATING BANDWIDTH OF THE TRANSMITTER

```
%Creation of the Interference Signal #1 (BPSK).
clear all;
A1=9*10^-2;  % Amplitude of the first Interference.
fs=93*10^6;   % Sampling rate of the card.
fc1=23.25*10^6; % Carrier Frequency of the first Interference.
Rb1=29.0625*10^5; % Bit Rate of the first Interference.
fchannel=80*10^6; % Operating Bandwidth of the Transmitter.
MaxBit1=1024;     % Number of bits of the first Interference.
SamplesPerBit1=fs/Rb1; %Samples per bit of the first Interference.
PeriodsPerSample1=fc1/Rb1;
fd1=fc1/fs;
i1=1-2*(rand(1,MaxBit1)<0.5); % Random Definition of symbols of the first Interference.
y1=[];
k1=reshape((0:MaxBit1*SamplesPerBit1-1),SamplesPerBit1,MaxBit1);
h=waitbar(0,'Modulating Signal 1 ...');
for bit=1:MaxBit1
        waitbar(bit/MaxBit1,h);
        y1=[y1,A1*i1(bit)*cos(2*pi*fd1*k1(:,bit))];%Transmitted signal of first interference
        y1=reshape(y1,1,MaxBit1*SamplesPerBit1);
end
close(h);
%Creation of the Interference Signal #2(BPSK).
A2=9*10^-2;     % The meaning of the symbols for the second Interference
fc2=fc1/2;          % as in the first one.
Rb2=Rb1/2;
MaxBit2=MaxBit1/2;
SamplesPerBit2=fs/Rb2;
PeriodsPerSample2=fc2/Rb2;
fd2=fc2/fs;
i2=1-2*(rand(1,MaxBit2)<0.5);
```

```
y2=[];
k2=reshape((0:MaxBit2*SamplesPerBit2-1),SamplesPerBit2,MaxBit2);
h=waitbar(0,'Modulating Signal 2 ...');
for bit=1:MaxBit2
        waitbar(bit/MaxBit2,h);
        y2=[y2,A2*i2(bit)*cos(2*pi*fd2*k2(:,bit))];
end
close(h);
y2=reshape(y2,1,MaxBit2*SamplesPerBit2);



% Interference plus AWGN.(FFT)
y12=y1+y2; % Sum of the two Interferences.
L=size(y12,2);
y12=y12+0.01*randn(1,L); % Sum of the two Interferences and AWGN.
spectrum=fft(y12);
psd=abs(spectrum).^2/(fs*L);
x=linspace(0,fs/2-1/size(psd,2), L/2);
maxPSD=max(psd);
logpsd=10*log10(psd);
df=fs/L;
Power=sum(df*psd);
figure
plot(x,logpsd(1:L/2)); % Gives the FFT of the 2 Interferences and the AWGN.
sTitle=sprintf('PSD of INTERFERENCE signal. Max: %5.2f dBW/Hz. Total Power: %5.2f dbW'...
                        ,10*log(maxPSD*df),10*log(Power));
title(sTitle);
xlabel('Frequency [Hz]');
grid on;



% Separation of the operating bandwidth of transmitter into n lower noise channels.
L80=round(L*fchannel/fs);% Because the operating bandwidth of the transmitter is 40MHz and the sampling
%frequency of the card is 93MHz, the points, corresponding to 40MHz, based on the fs=93MHz are found.
Qchannels=8; % Defines the number of the lower noise spectral areas (and by extension the number of the
%channels of QPUC) in which  the operating bandwidth of the transmitter is seperated.
QBW=3.5*10^6; %Defines the Bandwidth of each of the above spectral areas (and by extension the bandwidth
%of each signal of QPUC.)
Low_bound=3*10^6;% Defines the lower limit of the operating bandwidth of the transmitter.
QBP=ceil(L*QBW/fs);% Defines the number of points that correspond to QBW .
```

```
sTitle=sprintf('Finding channel noise power for %2.0d channels of BW=% 2.0d ...',Qchannels,QBW);
h=waitbar(0,sTitle);
kk=ceil(L*Low_bound/fs); %Defines the number of points that correspond to lower frequency,Low_bound .
fpoints=[];
QChP=[];
for f=kk:(L80/2)-QBP+1
        waitbar(f/(L80/2-QBP+1),h);
        QChP=[QChP,sum(psd(f:f+QBP-1))]; %The first channel starts at
        fpoints=[fpoints,f]; %the point 1 and stops at the point 1+QBP,the second  starts at the point 2 and
         end; %stops at the point 2+QBP etc. The QChP gives the noise power for all of these channels.
close(h);

figure
plot(fpoints,10*log10(QChP));%This plot gives the noise power versus initial point, for each of the above
%channels.
title('Channel noise power versus points');
grid on;

figure
plot(fpoints*fchannel/L80,10*log10(QChP));  %This plot gives the noise power versus initial frequency for each
%of the above channels.
title('Channel noise power versus frequency');
grid on;

QChpSort=sort(QChP);  %The above channels are sorted in ascending order.
Occupied=zeros(1, L80/2);

k=[];
h=waitbar(0,'Finding empty frequency bands ...');
for f=1:size(QChpSort,2)
        waitbar(f/size(QChpSort,2), h);
        K=find(QChP==QChpSort(f));
        k=[k,K];
end
close(h);

QChanStarts=[];
for n=1:size(k,2)
        if sum(Occupied(k(n)+kk-1:k(n)+kk-1+QBP-1))==0
        QChanStarts=[QChanStarts, k(n)+kk-1];          %This loop defines the lower noise areas in the
```

```
        Occupied(k(n)+kk-1:k(n)+kk-1+QBP-1)=ones(1,QBP); %operating bandwidth of the transmitter.
        end

if size(QChanStarts,2)==Qchannels
        break
end
end
```

```
%In order to understand better, how the lower noise areas in the
%operating bandwidth of the transmitter are defined (code lines 80-136), the below
%quite simple example is demonstrated, where the variables that are used have the same name as in
%the real code. Suppose that the variable L80 corresponds to 30 points, so L80/2=15 points .
%The variable kk let's be 5 and the QBP let's be 2.The code starts to
%calculate the noise power (the values are unreal) for each channel of QBP=2 points, starting from
%kk=5.The variable k gives, what is the position of the elements of matrix Qchpsort
%in matrix Qchp.
```

%Position in matrices %	Points (refered to each channel)	Qchp Matrix (gives noise power for each channel)	Qchpsort (sorts the Qchp matrix)	k
% 1	5,6	3	1	6
% 2	6,7	4	1	7
% 3	7,8	6	2	4
% 4	8,9	2	3	1
% 5	9,10	5	4	2
% 6	10,11	1	4	10
% 7	11,12	1	5	5
% 8	12,13	6	6	3
% 9	13,14	7	6	8
% 10	14,15	4	7	9

```
%Based on kk, k, QBP the positions of matrix Occupied are filled with
%ones. The below results are taken, applying the lines of code 126-128
%  for k=6 fill in positions 10,11,
```

```
%  for k=7 fill in no positions because position 11 is  reserved,
%  for k=4 fill in positions 8,9,
%  for k=1 fill in positions 5,6,
%  for k=2 fill in no positions because position 6 is reserved,
%  for k=10 fill in positions 14,15,
%  for k=5 fill in no positions because position 9 is reserved etc.

%Columns of Matrix Occupied          1  2  3  4  5  6  7  8  9  10  11  12  13  14  15
%filled by ones.                                    1  1     1  1    1   1    1    1   1   1
%
%The matrix QChanStarts gives the following results:QChanStarts=[10,8,5,14,12].

figure
plot(10*log10(psd(1:L80/2)));
title('The lower noise spectrum areas in the operating bandwidth of the transmitter.');
grid on;

for n=1:size(QChanStarts,2)
        line([QChanStarts(n); QChanStarts(n)+QBP-1],[-85,-85],'color','r','linewidth',4);
end

%Calculation of the noise power of the lower noise areas.
QNoisepower=[];
frequencyareas=[];
for wq=1:length(QChanStarts);
        o=QChP(QChanStarts(wq)-kk+1);
        QNoisepower=[QNoisepower,o]; %Calculation of the noise power of the lower noise areas.
        fr=(fs*QChanStarts(wq)/L);     % in the operating bandwidth of the transmitter.
        frequencyareas=[frequencyareas,fr]; %Estimation where the lower noise areas start in frequency.
end
```

B. MATLAB_CODE_2 (MAIN FILE)

```
% PERFORMANCE OF THE CARD IN AWGN
clear all;
fsclock=93*10^6;   % Clock Rate of the system.
Symbols1=15000; % Number of symbols for the signal.
k=2;          % For QPSK use k=2 for 8-PSK use k=3 and for 16-PSK k=4.
Rs1=2.90625/2*10^6;    % Symbol rate of Signal. It is determined based on the equations 11-13.
```

```
if (k==1)            % Initial phase of signal.
        InitialPhase=0;
else
        InitialPhase=pi/(2^k);
end;

PlotPSD=false; %Display diagrams.
PbTarget=10^-5; %Target probability of error.
MaxTrials=ceil(10/(Symbols1*PbTarget));% Defines the number of loops(below)
                                %that the program has to execute, for different
                                %each time noise power.
Asignal=1e-3; %Amplitude of the signal.
S=Asignal^2/2;%Power of the signal.
Eb=S/(k*Rs1);  %Eb of the signal in the input of the receiver.
                %It comes from the equation of SNR,i.e.S/N=Eb*Rb/N.
Pb=[];
EbNo=[];
Denom=0;
while   Denom<15            %It is referred to the number of dB for Eb/No. It has to increase
                            %when performance of 8-PSK,16-PSK, is investigated.
        No=Eb/10^(.1*Denom); %Instead of, in each set of loops to increase the power of the signal
                            %keeping the noise power fixed, the signal's power is kept fixed
                            %and  the power of noise is reduced. The equality comes from
                            %the relation 10*log(Eb/No)=Denom.
        sigma=sqrt(No*fsclock); %Square root of variance of noise.
                            %To be more specific: Assume that the channel
                            %with bandwidth fsclock
                            %corresponds to L discrete points.
                            %The frequency resolution is df=fsclock/L while the
                            %noise power for each discrete frequency is sigma^2/L
                            %(sigma^2 is the noise power for the whole spectrum)and
                            %the noise power per Hertz  i.e.
                            %the power spectral density is No=sigma^2/fsclock.
        TotalErrors=0;      %Initial value for errors.
        waitTitle=sprintf('Calculating BER for E_b/N_o=%2d dB',Denom);
        h=waitbar(0, waitTitle);

        for trial=1:MaxTrials  %Initialization of the loops.
                waitbar(trial/MaxTrials, h);
```

```
%SIGNAL
fc1=10^7;      %Carrier frequency of the transmitted signal.
IntFactor=4;  %Interpolation factor.
fsinitial=fsclock/IntFactor;    %Sampling frequency of signal prior interpolation.
data1=randint(1,Symbols1,2^k);  %Symbols that correspond to the signal.
SamplesPerSymbol1=round(fsinitial/Rs1);  %Samples per symbol.
DataMat1=[];
for i=1:SamplesPerSymbol1
DataMat1=[DataMat1; data1]; %Sampled Data.
end;
n1=reshape(0:Symbols1*SamplesPerSymbol1-1,SamplesPerSymbol1,Symbols1);
IBB=reshape(cos(2*pi*DataMat1/(2^k)+InitialPhase),1,Symbols1*SamplesPerSymbol1);
%I baseband Signal.
QBB=reshape(sin(2*pi*DataMat1/(2^k)+InitialPhase),1,Symbols1*SamplesPerSymbol1);
%Q baseband Signal.

if ((trial==MaxTrials) & PlotPSD)
PSDPlot(IBB, fsinitial, 'Baseband I channel Prior Interpolation', false);
end;


%WHY THE INTERPOLATED SIGNAL COMING FROM
%FUNCTIONS  “interpolate”  ”interpolatesrc” , IS MULTIPLIED BY 1.1*IntFactor

%In the case of interpolation by a factor IntFactor,
%the initial power of the signal
%reduces by IntFactor, since  more samples are added  between the initial
%samples. In order to maintain the power that signal had before
%the interpolation,  the interpolated signal, is multiplied by the
%interpolation factor,IntFactor. The factor 1.1 is empirical
%coming from the comparison of the signals' spectrums before and after
%interpolation.


%Interpolation using FIR filter. Reactivate these commands
%in order to use simple FIR filter instead of square root raised cosine FIR filter.
%The function “interpolate”, executes the interpolation and
%the filtering of the interpolated signal.
```

```
%IBBIntrpFiltr=1.1*IntFactor*interpolate(IBB, IntFactor, SamplesPerSymbol1, fsinitial*IntFactor, false);

%QBBIntrpFiltr=1.1*IntFactor*interpolate(QBB, IntFactor, SamplesPerSymbol1, fsinitial*IntFactor, false);


%Interpolation using FIR square root raised cosine

%filter till sampling frequency fsclock=93MHz.

FiltOrdInt=32;%This is the order of the filter.

%It has to change if  the interpolation

%factor has been changed.(See footnote 4 and table 6)

%The function “interpolatesrc”, executes the interpolation and

%the filtering of the interpolated signal.

IBBIntrpFiltr=1.1*IntFactor*interpolatesrc(IBB, IntFactor, FiltOrdInt, fsinitial*IntFactor, false);

QBBIntrpFiltr=1.1*IntFactor*interpolatesrc(QBB, IntFactor, FiltOrdInt, fsinitial*IntFactor, false);


if ((trial==MaxTrials) & PlotPSD)

PSDPlot(IBBIntrpFiltr, fsinitial*IntFactor, 'Baseband I channel after interpolation and filtering', false);

end;


%Transmission of interpolated signal.

n2=0:Symbols1*SamplesPerSymbol1*IntFactor-1;

TxData1=Asignal*IBBIntrpFiltr.*cos(2*pi*fc1/fsclock*n2)-
Asignal*QBBIntrpFiltr.*sin(2*pi*fc1/fsclock*n2);

%Transmitted I,Q Data at carrier frequency fc1.


if ((trial==MaxTrials) & PlotPSD)

PSDPlot(TxData1, fsclock, 'Transmitted Signal', false);

end;


%Received Data

TxInterAWGN=sigma*randn(1,length(TxData1)); %Creation of  AWGN in the channel.

RxData=TxData1+TxInterAWGN; %Received signal affected by  AWGN.
```

```
if ((trial==MaxTrials) & PlotPSD)
PSDPlot(RxData, fsclock, 'Received Signal', false);
end;


%Demodulation in baseband, where the I,Q baseband data are taken.
n3=0:length(RxData)-1;
I=RxData.*cos(2*pi*n3*fc1/fsclock+InitialPhase);
Q=-RxData.*sin(2*pi*n3*fc1/fsclock+InitialPhase);

if ((trial==MaxTrials) & PlotPSD)
PSDPlot(I+Q, fsclock, 'I Channel After the baseband demodulation', false);
end;


%CIC Decimation
M=1; %Number of delays.(Default value.)
N=5; %Number of stages.(Default value.)
R=4;%Smallest decimation factor, of CIC filter.
quant=quantizer;
Idec=cicdecimate(M,N,R,I,quant)/((R*M)^N);%Decimated I signal after  CIC filter.
Qdec=cicdecimate(M,N,R,Q,quant)/((R*M)^N);%Decimated Q signal after  CIC filter.


if ((trial==MaxTrials) & PlotPSD)
PSDPlot(Idec, fsclock/R, 'I Channel After the CIC filter', false);
end;


DecimationFactor=4; %Decimation Factor for the filter compute engine filter.
fsDecIn=fsclock/R;  %Sampling frequency of signal,
                    %after the first decimation imposed by CIC filter.
DecSamplesPerSymbolIn=round(SamplesPerSymbol1*IntFactor/R);%Samples per
                                                %symbol after CIC decimation.


% Decimation with LPF filter. Reactivate in order
%to use for decimation, a simple FIR filter.
%The function “decimate”, executes the filtering and
%the decimation of the signal, coming from  CIC filter.
```

```
%DecI=decimate(Idec, fsDecIn, DecimationFactor, Symbols1, DecSamplesPerSymbolIn, false);

%DecQ=decimate(Qdec, fsDecIn, DecimationFactor, Symbols1, DecSamplesPerSymbolIn, false);



%Decimation using FIR square root raised cosine filter

FiltOrdDec=32;%Defines the order of the filter. It has to change if we change the %decimation factor of FCE filter. (See footnote 4.)

%The function “decimatesrc”,executes the filtering and

%the decimation of the signal coming from CIC filter.

DecI=decimatesrc(Idec, fsDecIn, DecimationFactor, Symbols1, DecSamplesPerSymbolIn, FiltOrdDec, false);

DecQ=decimatesrc(Qdec, fsDecIn, DecimationFactor, Symbols1, DecSamplesPerSymbolIn, FiltOrdDec, false);


if ((trial==MaxTrials) & PlotPSD)

PSDPlot(DecI, fsclock/(R*DecimationFactor), 'I Channel After filtering and second decimation', false);

end;



%Integration of the remained samples per symbol, after the above two decimations, in

%order the receiver to decide about the received symbol.

BaseSamples=floor(SamplesPerSymbol1*IntFactor/(R*DecimationFactor));

Remainder=SamplesPerSymbol1*IntFactor/(R*DecimationFactor)-BaseSamples;

IntSamples=BaseSamples-1; % NOT all the samples per symbol take part in the integration.

Offset=1;

SumI=[];

SumQ=[];

j=0; % Initial Offset. During the loop it changes.

i=1; % Counter indicating the position in the matrix.

Remainder1=0;



%This loop has been created in order to encounter the case, where

%the remained for integration number of samples per symbol, is

%not integer. For example if it is 6.25, in the first

%three times six samples per symbol are integrated while in the

%fourth time seven samples per symbol , are integrated etc.

while i<length(DecI)

j=j+Remainder+Remainder1;
```

```
if j<Offset
SI=DecI(i:i+BaseSamples-1);
SQ=DecQ(i:i+BaseSamples-1);
SuI=sum(SI(:,BaseSamples-IntSamples+1:BaseSamples))/(IntSamples);
SuQ=sum(SQ(:,BaseSamples-IntSamples+1:BaseSamples))/(IntSamples);
i=i+BaseSamples;
else
SI=DecI(i:i+BaseSamples);
SQ=DecQ(i:i+BaseSamples);
SuI=sum(SI(:,BaseSamples-IntSamples+2:BaseSamples+1))/(IntSamples);
SuQ=sum(SQ(:,BaseSamples-IntSamples+2:BaseSamples+1))/(IntSamples);
i=i+BaseSamples+1;
Remainder1=j-Offset;
j=0;
end
SumI=[SumI,SuI];
SumQ=[SumQ,SuQ];
end



%Calculate phases and correct
RxPhi=atan2(SumQ,SumI);%From I,Q, the phase, using the atan2 command
                       %(and NOT simply the atan, because it
                       %can't give all the phases) is derived.
RxPhi(find(RxPhi<0))=RxPhi(find(RxPhi<0))+2*pi;% When the phase is negative
                                               % add 2*pi.
RxPhi=round(RxPhi*(2^(k-1))/pi);%From the phase, the symbols are extacted,
                                %based on the equation:
                                %Phase=2*pi*symbol/2^k
RxPhi(find(RxPhi==(2^k)))=0;%Replace symbols 2^k by 0.

%Calculate Errors
Errors=sum(data1~=RxPhi); % Calculates the difference between the transmitted
                          % and the received symbols during each loop.
TotalErrors=TotalErrors+Errors;% Adds the errors coming from each loop.
if TotalErrors>500
break
end;
end;
```

```
            close(h);
            Symbols1*trial
            TotalErrors
            BER=TotalErrors/(Symbols1*trial)%Calculates the symbol error rate, dividing the total errors
                                                %by the total number of symbols, transmitted after the
                                                %completion of the number of the  loops, that correspond to variable
                                                %maxtrial.
            Pb=[Pb,BER/k];              %Calculates the BER.
            EbNo=[EbNo,Denom];
            Denom=Denom+1; %Increases the variable Denom. In this way, AWGN is reduced.
                                %Execution of loop, gives a new BER,
                                %for this reduced value of noise. It is obvious that as Denom
                                %increases, AWGN power is reduced and by extension  the
                                %BER is reduced.
end;
figure
semilogy(EbNo,Pb,'b');
ylabel('Pb');
xlabel('E_b/N_o');
grid;
hold on;                    %Plots the theoretical vs. the performance of the system
                            %according to the parameters defined above.
EbNoTheor=1:0.5:100;
PbTheor=1/k*erfc(sqrt(k*EbNoTheor)*sin(pi/2^k));
semilogy(10*log10(EbNoTheor),PbTheor,'xr');
axis([0,25,10^-6,10^0]);
title('Performance of the System for different number of samples vs Theoretical Performance.')
grid;
```

C. MATLAB_CODE_2 (FUNCTIONS)

FUNCTION INTERPOLATE

```
function [SignalOut] = interpolate(SignalIn, IntFactor, SamplesPerSymbol, fsHigh, displayFilter)

%Interpolates the input signal adding zeros between samples.
IntSignal=[SignalIn;zeros(IntFactor-1,length(SignalIn))];
IntSignal=reshape(IntSignal,1,IntFactor*length(IntSignal));

%Creation of the simple FIR Filter that will filter the interpolated signal.
rp = 1;         % Passband ripple
```

```
rs = 60;        % Stopband ripple
fint = [fsHigh/(4*IntFactor) fsHigh/(2*IntFactor)]   %Pass band-Cutoff frequencies.
                                                     %Each time that the interpolation factor
                                                     %changes, they change.
a = [1 0];      % Desired amplitudes
% Compute deviations
dev = [(10^(rp/20)-1)/(10^(rp/20)+1) 10^(-rs/20)];
[nInt,fo,ao,w] = remezord(fint,a,dev,fsHigh);
LPF = remez(nInt,fo,ao,w) %Defines the coefficients and the order of the filter.
                                %They change each time  the interpolation factor is changed.

if displayFilter
        fvtool(LPF); %Displays  the transfer function of the FIR filter.
end

%Filters the  interpolated signal.
IntFiltSignal=conv(LPF,IntSignal);

%The below loop detracts from IntFiltSignal the points that correspond to
%the delays of the filter, which are N/2 points from the start and N/2
%points from the end of IntFiltSignal, when N is even and (N+1)/2
%when N is odd. N is the order of the filter.
if mod(nInt,2)~=0
        g=(nInt+1)/2;
        IntFiltSignal=IntFiltSignal(g+1:length(IntFiltSignal)-g+1);
else
        g=nInt/2;
        IntFiltSignal=IntFiltSignal(g+1:length(IntFiltSignal)-g);
end

%SignalOut=reshape(IntFiltSignal,SamplesPerSymbol*IntFactor,length(IntFiltSignal)/(SamplesPerSymbol*IntFa
ctor));
SignalOut=IntFiltSignal;
```

FUNCTION INTERPOLATESRC

```
function [SignalOut] =interpolatesrc(SignalIn, IntFactor, FltrOrder, fsHigh, displayFilter)

%The below two lines execute the interpolation, adding zeros between samples
```

```
%and give the interpolated signal “IntSignal”.
IntSignal=[SignalIn;zeros(IntFactor-1,length(SignalIn))];
IntSignal=reshape(IntSignal,1,IntFactor*length(IntSignal));

%Creation of square root raised cosine filter which filters the signal
%after the interpolation.
Fcutoff=fsHigh/(2*IntFactor);   %Defines the cutoff frequency of filter,with fsHigh=fsclock
df=fsHigh/(2*IntFactor)-fsHigh/(4*IntFactor); %Defines the transition region of filter
LPFsrc=firrcos(FltrOrder,Fcutoff,df,fsHigh,'sqrt'); %Defines the coefficients of the square root
                                                    %raised cosine FIR filter.
                                                    %The order of the filter is user defined.
                                                    %The same number of coefficients is used, with
                                                    %that derived from the function interpolate
                                                    %when a simple FIR filter is used.
                                                    %Each time the interpolation factor
                                                    %changes, the cutoff frequency and the transition
                                                    region change.
if displayFilter
        fvtool(LPFsrc) %Gives the transfer function of the filter.
end



%Filtering of interpolated signal
IntFiltSignal=conv(LPFsrc,IntSignal); %Signal passes through filter.

%The below loop detracts from IntFiltSignal the points that correspond to
%the delays of the filter, which are N/2 points from the start and N/2
%points from the end of IntFiltSignal, when N is even and (N+1)/2
%when N is odd. N is the order of the filter.
if mod(FltrOrder,2)~=0
        g=(FltrOrder+1)/2;
        IntFiltSignal=IntFiltSignal(g+1:length(IntFiltSignal)-g+1);
else
        g=FltrOrder/2;
        IntFiltSignal=IntFiltSignal(g+1:length(IntFiltSignal)-g);
end

SignalOut=IntFiltSignal;%Output of function interpolatesrc.
```

FUNCTION DECIMATE

```
function [SignalOut]=decimate(SignalIn, fsIn, DecFactor, Symbols, SamplesPerSymbolIn, displayFilter)

%FIR Filter design
rp = 1;         % Passband ripple
rs = 60;        % Stopband ripple
fdec = [fsIn/(4*DecFactor) fsIn/(2*DecFactor)]   %Passband-Cutoff frequencies. Each time the
                                                 %decimation factor or the sampling frequency
                                                 %after the CIC filter changes, they change.
a = [1 0];      % Desired amplitudes
% Compute deviations
dev = [(10^(rp/20)-1)/(10^(rp/20)+1) 10^(-rs/20)];
[ndec,fo,ao,w] = remezord(fdec,a,dev,fsIn);
LPF = remez(ndec,fo,ao,w) %Defines the order and the coefficients of the filter.
                          %They change each time the decimation factor is changed.
if displayFilter
        fvtool(LPF); %Displays the filter's transfer function.
end

%Filters the signal prior to decimation.
filteredSignal=conv(LPF,SignalIn);

%The below loop detracts from filteredSignal the points that correspond to
%the delays of the filter, which are N/2 points from the start and N/2
%points from the end of filteredSignal, when N is even and (N+1)/2
%when N is odd. N is the order of the filter.
if mod(ndec,2)~=0
        g=(ndec+1)/2;
        filteredSignal=filteredSignal(g+1:length(filteredSignal)-g+1);
else
        g=ndec/2;
        filteredSignal=filteredSignal(g+1:length(filteredSignal)-g);
end

%Decimation of the filtered signal
decimatedSignal=reshape(filteredSignal,DecFactor,Symbols*SamplesPerSymbolIn/DecFactor);
SignalOut=decimatedSignal(1,:);%The output of the function is the decimated signal.
```

FUNCTION DECIMATESRC

```
function [SignalOut]=decimatesrc(SignalIn, fsIn,DecFactor, Symbols, SamplesPerSymbolIn, FltrOrder, displayFilter)


%Creation of square root raised cosine filter, which will filter the signal
%prior to decimation.
Fcutoff=fsIn/(2*DecFactor)%Defines the Cutoff Frequency of the filter using
                          %as sampling frequency, the sampling frequency that signal
                          %has, after the decimation imposed by CIC filter.

df=fsIn/(2*DecFactor)-fsIn/(4*DecFactor) %Defines the transition Region of the filter.
LPFsrc=firrcos(FltrOrder,Fcutoff,df,fsIn,'sqrt');%Defines the coefficients of the square root
                                                 %raised cosine FIR filter. The order of the filter is user defined.
                                                 %The number  of coefficients  is the same with
                                                 %that, derived from the function decimate,
                                                 %when prior to decimation a simple FIR filter is used.
                                                 %Each time the decimation factor or
                                                 %the sampling frequency after CIC filtering
                                                 %changes, the cutoff frequency and the transition region
                                                 %change.

if displayFilter
        fvtool(LPFsrc);%Gives the transfer function of the filter.
end

%filtering of signal
filteredSignal=conv(LPFsrc,SignalIn);%Signal passes through the filter.

%The below loop detracts from filteredSignal the points that correspond to
%the delays of the filter, which are N/2 points from the start and N/2
%points from the end of filteredSignal, when N is even and (N+1)/2
%when N is odd. N is the order of the filter.
if mod(FltrOrder,2)~=0
        g=(FltrOrder+1)/2;
        filteredSignal=filteredSignal(g+1:length(filteredSignal)-g+1);
else
        g=FltrOrder/2;
```

```
        filteredSignal=filteredSignal(g+1:length(filteredSignal)-g);
end


%The below two lines execute the decimation, detracting samples from the initial samples
%of the signal and, give the decimated signal "SignalOut".
decimatedSignal=reshape(filteredSignal,DecFactor,Symbols*SamplesPerSymbolIn/DecFactor);
SignalOut=decimatedSignal(1,:);



FUNCTION PSDPLOT
function []=PSDPlot(SignalIn, fs, sTitle, normalized)
spectrum=fft(SignalIn);
L=max(size(SignalIn,2), size(SignalIn,1));
SignalPSD=abs(spectrum).^2/(fs*L);
AvgPSD=sum(SignalPSD)/L;
Power=AvgPSD*fs;
x=linspace(0,fs/2-1/size(SignalPSD,2), L/2);
maxPSD=max(SignalPSD);
if normalized
        logpsd=10*log10(SignalPSD/maxPSD);
else
        logpsd=10*log10(SignalPSD);
end
figure
plot(x,logpsd(1:L/2));
sTitle2=sprintf('\nPower: %6.1f dbW. Average PSD: %6.1f dBW. Max PSD: %6.1f dBW/Hz'...
                ,10*log10(Power),10*log10(AvgPSD),10*log10(maxPSD));
sTitle=['Power Spectral Density of the ', sTitle, sTitle2];
title(sTitle);
xlabel('Frequency');
ylabel('Power Spectral Density [dBW]');
grid on;
```

D. MATLAB_CODE_3 (MAIN FILE)

```
%PERFORMANCE OF THE CARD FOR DIFFERENT RATIOS OF J/S

clear all;
```

```
fsclock=93*10^6;   % Clock Rate of the system.
fc=30*10^6;     % Frequency of signal.
Symbols=24000;  % Number of Symbols for signal.
Symbols1=24000/2; % Number of symbols for the first interference.
Symbols2=24000*(2/3); % Number of symbols for the second interference.
k=2;            % For QPSK use k=2 for 8-PSK use k=3 and for 16-PSK k=4.
Rs=1.453125*10^6;     % Symbol Rate of signal. It is determined based on the equations 11-13.
Rs1=1.453125*10^6/2;    % Symbol rate of the first interference
Rs2=1.453125*10^6*(2/3);    % Symbol Rate of the second interference
if (k==1)
            InitialPhase=0;
else
            InitialPhase=pi/(2^k);
end;

PlotPSD=false; %Display diagrams
PbTarget=10^-5; %Target probability of error
MaxTrials=ceil(11/(Symbols*PbTarget));% Defines the number of loops(below)
                                    %that program has to execute, for a specific position of
                                    %interferences in relation to signal and, for different
                                    %each time noise power.
Aintf1=1e-3;      %Amplitude of first interference .It is referred to the input of the receiver.
Aintf2=1e-3;      %Amplitude of second interference. It is referred to the input of the receiver.
Asignal=1e-3;     %Amplitude of signal. It is referred to the input of the receiver.
S=Asignal^2/2;   %Average power of the signal's carrier in the input of the receiver.
Eb=S/(k*Rs);      %It comes from the equation of SNJR, i.e.S/(N+J)=Eb*Rb/(N+J).
Pb=[];
EbNo=[];
Denom=0;
JSRatio=[];

% INTERFERENCE #1.It is a 8-FSK signal.
fc1=20*10^6;   %Carrier frequency of the first interference.
data1=randint(1,Symbols1,2^3); %Data of the first interference.
TxInter1=Aintf1*dmod(data1,fc1,Rs1,fsclock,'FSK',2^3); %Transmitted signal of the first interference.

% INTERFERENCE #2.It is a QPSK signal.
fc2=40*10^6;  %Carrier frequency for the second interference.
data2=randint(1,Symbols2,2^2); %Data for the second interference.
SamplesPerSymbol2=round(fsclock/Rs2); %Defines samples per symbol.
```

```
n2=reshape(0:Symbols2*SamplesPerSymbol2-1,SamplesPerSymbol2,Symbols2);
DataMat2=[];
for i=1:SamplesPerSymbol2
DataMat2=[DataMat2; data2]; %Sampled data.
end;
TxInter2=reshape(Aintf2*sin(2*pi*fc2/fsclock*n2+2*pi*DataMat2/(2^k)+pi/4),1,Symbols2*SamplesPerSymbol2
);    %Transmitted signal of the second interference.

TxInter=TxInter1+TxInter2; %Transmitted Interference.
spectrum=fft(TxInter);     %Power of interference.
L=length(TxInter);
InterPSD=abs(spectrum).^2/(fsclock*L);
J=sum(InterPSD(round(L*fc/fsclock-2*L*Rs/fsclock):round(L*fc/fsclock+2*L*Rs/fsclock)));
                                %Power of interference in the bandwidth of the transmitted
                                %signal. Since only the main and the two side lobes are of interest, this
                                %bandwidth is equal to 4*Rs where Rs is the symbol rate of signal.
%In order to increase the power of interference that falls into the bandwidth of signal, deriving in this way
%a different curve J/S , the carrier frequencies fc1,fc2 has to be changed each time the code is executed.

while   Denom<15                %It is referred to the number of dB for Eb/No. It has to increase
                                %when  8-PSK,16-PSK signals are used.
          No=Eb/10^(.1*Denom);  %Instead of, in each set of loops to increase the power of the signal
                                %keeping the noise power fixed, the signal's power is kept fixed
                                %and the power of noise is reduced. The equality comes from
                                %the relation 10*log(Eb/No)=Denom
          sigma=sqrt(No*fsclock);   %Square root of variance of noise.
                                %To be more specific: Suppose that  channel
                                %with bandwidth fsclock
                                %corresponds to L discrete points.
                                %The frequency resolution is df=fsclock/L while the
                                %noise power for each discrete frequency is sigma^2/L
                                %(sigma^2 is the noise power for the whole spectrum)and
                                %the noise power per Hertz  i.e.
                                %the power spectral density is No=sigma^2/fsclock.
          TotalErrors=0;    %Initial value for errors.
          waitTitle=sprintf('Calculating BER for E_b/N_o=%2d dB',Denom);
          h=waitbar(0, waitTitle);

          for trial=1:MaxTrials
                    waitbar(trial/MaxTrials, h);
```

```
if ((trial==MaxTrials) & PlotPSD)
PSDPlot(TxInter1, fsclock, '8FSK INTERFERENCE', false);
end;

if ((trial==MaxTrials) & PlotPSD)
PSDPlot(TxInter2, fsclock, '16PSK INTERFERENCE', false);
end;

if ((trial==MaxTrials) & PlotPSD)
PSDPlot(TxInter, fsclock, 'INTERFERENCES', false);
end;

%SIGNAL
fc=30*10^6; %Carrier frequency of the transmitted signal.
IntFactor=4; %Interpolation factor.
fsinitial=fsclock/IntFactor;%Sampling frequency of signal prior interpolation.
data=randint(1,Symbols,2^k); %Data of signal.
SamplesPerSymbol=round(fsinitial/Rs);%Samples per symbol prior to interpolation.
DataMat=[];
for i=1:SamplesPerSymbol
DataMat=[DataMat; data]; %Sampled Data of signal.
end;
n=reshape(0:Symbols*SamplesPerSymbol-1,SamplesPerSymbol,Symbols);
IBB=reshape(cos(2*pi*DataMat/(2^k)+InitialPhase),1,Symbols*SamplesPerSymbol);%I
%baseband Signal.
QBB=reshape(sin(2*pi*DataMat/(2^k)+InitialPhase),1,Symbols*SamplesPerSymbol);%Q
%baseband Signal.

if ((trial==MaxTrials) & PlotPSD)
PSDPlot(IBB, fsinitial, 'Baseband I channel Prior Interpolation', false);
end;


%WHY  THE INTERPOLATED SIGNAL COMING FROM  FUNCTION
%"interpolatesrc", IS MULTIPLIED BY 1.1*IntFactor.

%In the case of interpolation by a factor IntFactor, the initial power of the signal
%reduces by IntFactor, since more samples between the initial
%samples are added. In order to maintain the power that, signal had before
```

```
%the interpolation, the interpolated signal is multiplied by the
%interpolation factor,IntFactor. The factor 1.1 is empirical
%coming from the comparison of the signals' spectrums before and after
%interpolation.

%Interpolation using FIR square root raised cosine filter
FiltOrdInt=32;%This is the order of the filter. It has to change when
                %the interpolation factor changes.(See footnote 4).
%The function “interpolatesrc”, executes the interpolation and
%the filtering of the interpolated signal.
IBBIntrpFiltr=1.1*IntFactor*interpolatesrc(IBB, IntFactor, FiltOrdInt, fsinitial*IntFactor, false);
QBBIntrpFiltr=1.1*IntFactor*interpolatesrc(QBB, IntFactor, FiltOrdInt, fsinitial*IntFactor, false);


if ((trial==MaxTrials) & PlotPSD)
PSDPlot(IBBIntrpFiltr, fsinitial*IntFactor, 'Baseband I channel after interpolation and filtering', false);
end;

%Transmission of interpolated signal with sampling frequency fsclock=93MHz.
n=0:Symbols*SamplesPerSymbol*IntFactor-1;
TxData=Asignal*IBBIntrpFiltr.*cos(2*pi*fc/fsclock*n)-
Asignal*QBBIntrpFiltr.*sin(2*pi*fc/fsclock*n);
%Transmitted I,Q Data at carrier frequency fc.

if ((trial==MaxTrials) & PlotPSD)
PSDPlot(TxData, fsclock, 'Transmitted Signal', false);
end;

%Received Data. It is the signal in the input of the receiver, affected by the interferences and the AWGN.
RxData=TxData+TxInter+sigma*randn(1,length(TxData));

if ((trial==MaxTrials) & PlotPSD)
PSDPlot(RxData, fsclock, 'Received Signal', false);
end;

spectrum=fft(TxData); %Power of transmitted signal.
L=length(TxData);
```

```
TxDataPSD=abs(spectrum).^2/(fsclock*L);
S=sum(TxDataPSD(round(L*fc/fsclock-
2*L*Rs/fsclock):round(L*fc/fsclock+2*L*Rs/fsclock)));
        %Power of signal in its bandwidth. Since only
        %the main and the two side lobes are of interest, this bandwidth is equal to 4*Rs
        where Rs is the symbol rate of the signal.
JSR=10*log10(J/S);
JSRatio=[JSRatio,JSR]; %Calculation of the ratio J/S in the bandwidth of signal.


%Demodulation in baseband
n4=0:length(RxData)-1;
I=RxData.*cos(2*pi*n4*fc/fsclock+InitialPhase);
Q=-RxData.*sin(2*pi*n4*fc/fsclock+InitialPhase);


if ((trial==MaxTrials) & PlotPSD)
PSDPlot(I, fsclock, 'I Channel After the baseband demodulation', false);
end;


%CIC Decimation
M=1;%Number of delays.(Default value).
N=5;%Number of stages.(Default value).
R=4;%Smallest decimation factor, of CIC filter.
quant=quantizer;
Idec=cicdecimate(M,N,R,I,quant)/((R*M)^N);%Decimated I signal after  CIC filter.
Qdec=cicdecimate(M,N,R,Q,quant)/((R*M)^N);%Decimated Q signal after  CIC filter.


if ((trial==MaxTrials) & PlotPSD)
PSDPlot(Idec, fsclock/R, 'I Channel After the CIC filter', false);
end;


DecimationFactor=4; %Decimation factor for the FCE.
fsDecIn=fsclock/R;  %Sampling Frequency of signal after the first decimation imposed by
%the CIC filter.
DecSamplesPerSymbolIn=round(SamplesPerSymbol*IntFactor/R);%Samples
%per symbol after CIC decimation.


%Decimation using FIR square root raised cosine filter.
FiltOrdDec=32;%Defines the order of the filter. It has to change when
                %the decimation factor of FCE filter changes.(See footnote 4.)
%The function “decimatesrc”, executes the filtering and
%the decimation of the signal coming from the CIC filter.
```

```
DecI=decimatesrc(Idec, fsDecIn,  DecimationFactor, Symbols, DecSamplesPerSymbolIn, FiltOrdDec, false);

DecQ=decimatesrc(Qdec, fsDecIn,  DecimationFactor, Symbols, DecSamplesPerSymbolIn, FiltOrdDec, false);


if ((trial==MaxTrials) & PlotPSD)

PSDPlot(DecI, fsclock/(R*DecimationFactor), 'I Channel  After Decimation', false);

end;


%Integration of the remained samples per symbol in order the receiver to decide for the %received symbol.

BaseSamples=floor(SamplesPerSymbol*IntFactor/(R*DecimationFactor));

Remainder=SamplesPerSymbol*IntFactor/(R*DecimationFactor)-BaseSamples;

IntSamples=BaseSamples-1; %Not all the remained samples per symbol,take place in the %integration.

Offset=1;

SumI=[];

SumQ=[];

j=0; % Initial Offset. During the loop it changes.

i=1; % Counter for the position of matrix.

Remainder1=0;


% This loop has been created, in order to encounter the case, where

%the remained for integration number of samples per symbol, is

%not integer. For example if it is 6.25,in the first

%three times  six samples per symbol have to be integrated, while in the

%fourth time  seven samples per symbol, etc.

while i<length(DecI)

j=j+Remainder+Remainder1;

if j<Offset

SI=DecI(i:i+BaseSamples-1);

SQ=DecQ(i:i+BaseSamples-1);

SuI=sum(SI(:,BaseSamples-IntSamples+1:BaseSamples))/(IntSamples);

SuQ=sum(SQ(:,BaseSamples-IntSamples+1:BaseSamples))/(IntSamples);

i=i+BaseSamples;

else

SI=DecI(i:i+BaseSamples);

SQ=DecQ(i:i+BaseSamples);

SuI=sum(SI(:,BaseSamples-IntSamples+2:BaseSamples+1))/(IntSamples);

SuQ=sum(SQ(:,BaseSamples-IntSamples+2:BaseSamples+1))/(IntSamples);

i=i+BaseSamples+1;
```

```
            Remainder1=j-Offset;
            j=0;
            end
            SumI=[SumI,SuI];
            SumQ=[SumQ,SuQ];
            end



            %Calculate phases and correct, as in case of AWGN.
            RxPhi=atan2(SumQ,SumI);%The phase is derived by the I,Q using the atan2 command
                                    %and NOT  the atan, because it can't give all the phases.
            RxPhi(find(RxPhi<0))=RxPhi(find(RxPhi<0))+2*pi;% When the phase is negative
                                                    % add 2*pi.
            RxPhi=round(RxPhi*(2^(k-1))/pi);%The symbols are extracted by the phase
                                        %based on the equation Phase=2*pi*symbol/2^k.
            RxPhi(find(RxPhi==(2^k)))=0;% Replace symbols 2^k by 0.


            %Calculate Errors
            Errors=sum(data~=RxPhi);%Calculates the difference between the transmitted
                    %and the received symbols during each loop.
            TotalErrors=TotalErrors+Errors;% Adds the errors coming from each loop.
            if TotalErrors>500
            break
            end;
end;


close(h);
Symbols1*trial
TotalErrors
JSR
BER=(TotalErrors/(Symbols1*trial))/k% Calculates the bit error rate, dividing the total errors
                                    %by the total number of symbols that transmitted after the
                                    %completion of the number of loops that correspond to maximum
                                    %value of variable maxtrial.
Pb=[Pb,BER];
EbNo=[EbNo,Denom];
Denom=Denom+1; %Increase the variable Denom, for a specific each time, position
                    %of the signal in relation to the interferences.
                    %In this way  the AWGN is reduced.
                    %Repeating the loops, a new BER is derived
```

```
                    %for this reduced value of noise. It is obvious that as Denom
                    %increases the AWGN power reduces and by extension  the
                    %BER reduces.

end;
meanJSR=sum(JSRatio)/length(JSRatio)     %The mean J/S is calculated because between the loops it changes %slightly. The order of these fluctuations is 10^-2.
figure
semilogy(EbNo,Pb,'r');
ylabel('Pb');
xlabel('E_b/N_o');
grid;
hold on;


EbNoTheor=1:0.5:100;
PbTheor=1/k*erfc(sqrt(k*EbNoTheor)*sin(pi/2^k));
semilogy(10*log10(EbNoTheor),PbTheor,'xr');
axis([0,35,10^-6,10^0]);
title('Performance of the System for various Ratios of J/R')
grid;
```

E. MATLAB_CODE_3 (FUNCTIONS)

The functions Interpolatesrc,Decimatesrc,PSDPlot as in MATLAB_CODE_2

F. MATLAB_CODE_4 (MAIN FILE)

```
%    OBTAINING OPTIMUM Rb BY POWER CONTROL


%Creation of the Interference Signal #1 (BPSK.)
clear all;
A1=sqrt(10^-3);  %Amplitude of the first Interference.
fsclock=93*10^6;  %Sampling rate of the card.
fc1=30*10^6; %Carrier Frequency of the first Interference.
Rb1=3*10^6;  %Bit rate of the first interference.
fchannel=80*10^6; % Operating Bandwidth of the Transmitter.
MaxBit1=2048;     % Number of bits of the first Interference.
SamplesPerBit1=fsclock/Rb1; %Samples per bit of the first Interference. Try mod(fsclock,Rb1)=0 in order to be %able to add same points in FFT.
PeriodsPerSample1=fc1/Rb1;
```

```
fd1=fc1/fsclock;
i1=1-2*(rand(1,MaxBit1)<0.5); %Random Definition of symbols of the first Interference.
y1=[];
k1=reshape((0:MaxBit1*SamplesPerBit1-1),SamplesPerBit1,MaxBit1);
h=waitbar(0,'Modulating Signal 1 ...');
for bit=1:MaxBit1
        waitbar(bit/MaxBit1,h);
        y1=[y1,A1*i1(bit)*cos(2*pi*fd1*k1(:,bit)+pi/4)];
end
close(h);
y1=reshape(y1,1,MaxBit1*SamplesPerBit1);%Transmitted signal of first interference.


%Creation of the Interference Signal #2 (BPSK.)
A2=sqrt(10^-3);      % The meaning of the symbols  for the second Interference
fc2=fc1/2;           % as in the first one.
Rb2=Rb1/2;
MaxBit2=MaxBit1/2;
SamplesPerBit2=fsclock/Rb2;
PeriodsPerSample2=fc2/Rb2;
fd2=fc2/fsclock;
i2=1-2*(rand(1,MaxBit2)<0.5);
y2=[];
k2=reshape((0:MaxBit2*SamplesPerBit2-1),SamplesPerBit2,MaxBit2);
h=waitbar(0,'Modulating Signal 2 ...');
for bit=1:MaxBit2
        waitbar(bit/MaxBit2,h);
        y2=[y2,A2*i2(bit)*cos(2*pi*fd2*k2(:,bit)+pi/2)];
end
close(h);
y2=reshape(y2,1,MaxBit2*SamplesPerBit2);


%Creation of the Interference Signal #3(BPSK.).
A3=sqrt(2*10^-3);      % The meaning of the symbols  for the third Interference
fc3=fc1/3;             % as in the first one.
Rb3=Rb1/2;
MaxBit3=MaxBit1/2;
SamplesPerBit3=fsclock/Rb3;
PeriodsPerSample3=fc3/Rb3;
```

```
fd3=fc3/fsclock;
i3=1-2*(rand(1,MaxBit3)<0.5);
y3=[];
k3=reshape((0:MaxBit3*SamplesPerBit3-1),SamplesPerBit3,MaxBit3);
h=waitbar(0,'Modulating Signal 3 ...');
for bit=1:MaxBit3
        waitbar(bit/MaxBit3,h);
        y3=[y3,A3*i3(bit)*cos(2*pi*fd3*k3(:,bit)+pi/12)];
end
close(h);
y3=reshape(y3,1,MaxBit3*SamplesPerBit3);


%Interference plus AWGN.(FFT)
y12=y1+y2+y3; % Sum of the Interferences.
L=size(y12,2);
No=10^-12; %PSD of AWGN
sigma=sqrt(No*fsclock);
ynoise=sigma*randn(1,L);%Defines AWGN in the channel.
y12=y12+ynoise; %Sum of  Interferences and  AWGN
                %in the operating bandwidth of the transmitter.
spectrum=fft(y12);
psd=abs(spectrum).^2/(fsclock*L);
x=linspace(0,fsclock/2-1/size(psd,2), L/2);
maxPSD=max(psd);
logpsd=10*log10(psd);
df=fsclock/L;
Power=sum(df*psd);
figure
plot(x,logpsd(1:L/2)); %Gives the FFT of  Interferences and  AWGN.
sTitle=sprintf('PSD of INTERFERENCE signal. Max: %5.2f dBW/Hz. Total Power: %5.2f dbW'...
                ,10*log(maxPSD*df),10*log(Power));
title(sTitle);
xlabel('Frequency [Hz]');
grid on;

%At this point starts the procedure of finding the optimum solution. In each
%loop we desire the noise plus interference to remain unchanged, because the channel is stationary.
BER=1;%Initial value of BER in order to start the loop.
BER_FINAL=[];
```

```
BEST_RESULT_STORY=[];
IntFactor=6;
DecimationFactor=6;
RCIC=4; %Decimation factor of CIC filter. It is fixed.(page 35)
while BER>1e-5;
DecimationFactor=2+DecimationFactor;%Begin with decimation factor
                                        %after the CIC filter equal to 8.
IntFactor=2+IntFactor;%Empirically this scheme gives the less errors.


%Separation of the operating bandwidth of transmitter into n lower noise spectral areas.
L80=round(L*fchannel/fsclock);%Because the operating bandwidth
%of the transmitter is 40MHz and the sampling frequency
%of the card is 93MHz  the points corresponding to 80MHz are found, based on  fs=93MHz.
Qchannels=8; %Defines the number of the lower noise spectral areas
%(and by extension the number of the channels of QPUC)
bw=93*10^6/(RCIC*DecimationFactor); %Defines the Bandwidth of each of the above spectral areas.
                                        %It is derived by equations 11-13.
Low_bound=3*10^6;%Defines the lower limit in the operating bandwidth of the transmitter.
QBP=round(L*bw/fsclock);%Defines the number of points that corresponds to bw .
sTitle=sprintf('Finding channel noise power for BW %2.0d channels of %2.0d BW...',Qchannels,bw);
h=waitbar(0,sTitle);
kk=round(L*Low_bound/fsclock); %Defines the number of points
                                    %that corresponds to lower frequency  (Low_bound .)
fpoints=[];
QChP=[];
for f=kk:(L80/2)-QBP+1
        waitbar(f/(L80/2-QBP+1),h);
        QChP=[QChP,sum(psd(f:f+QBP-1))]; %The first channel starts at
        fpoints=[fpoints,f];                    % point kk and stops at  point kk+QBP-1,
                                                %the second channel starts at  point kk+1 and
end;                                            %stops at  point kk+1+QBP-1 e.t.c.
                                                % The QChP gives the noise power for all of these channels.
close(h);

figure
plot(fpoints,10*log10(QChP));%This plot gives the noise power versus
                                %initial point for each of the above channels.
title('Channel noise power versus points');
grid on;
```

```
figure
plot(fpoints*fchannel/L80,10*log10(QChP));  %This plot gives the noise
                                            %power versus starting frequency,
title('Channel noise power versus frequency'); %for each one of the above channels.
grid on;

QChpSort=[];
QChpSort=sort(QChP);  % The above channels are sorted in ascending order
                      %(from less to noisiest channel.)
Occupied=[];
Occupied=zeros(1, L80/2);
k=[];
h=waitbar(0,'Finding empty frequency bands ...');
for f=1:size(QChpSort,2)
        waitbar(f/size(QChpSort,2), h);
        K=find(QChP==QChpSort(f));
        k=[k,K];
end
close(h);

QChanStarts=[];
for n=1:size(k,2)
        if sum(Occupied(k(n)+kk-1:k(n)+kk-1+QBP-1))==0
        QChanStarts=[QChanStarts, k(n)+kk-1];                 %This loop defines the lower noise areas in the
        Occupied(k(n)+kk-1:k(n)+kk-1+QBP-1)=ones(1,QBP); %operating bandwidth
                                                          %of the transmitter by filling with ones
                                                          %the matrix Occupied. More
                                                          %details in Matlab_code_1.
        end

if size(QChanStarts,2)==Qchannels %Gives the exact number of predefined
                                  %lower noise spectral areas.
        break
end
end

figure
plot(10*log10(psd(1:L80/2)));
title('The lower noise spectrum areas in the operating bandwidth of the transmitter.');
```

```
grid on;

for n=1:size(QChanStarts,2)
        line([QChanStarts(n); QChanStarts(n)+QBP-1],[-85,-85],'color','r','linewidth',4);
end

%Calculation of  noise power and  starting frequencies of the lower noise areas.
QNoisepower=[];
frequencyareas=[];
for wq=1:length(QChanStarts);
        o=QChP(QChanStarts(wq)-kk+1);
        QNoisepower=[QNoisepower,o]; %Calculation of  noise
                                     %power of  lower noise areas. This power is referred
                                     %to AWGN and  power coming from  interference signals.
        fr=fsclock*QChanStarts(wq)/L;
        frequencyareas=[frequencyareas,fr]; %Calculates where the lower noise
                                            % areas start in frequency.
end

%Calculates the best combination (number of channels, modulation scheme),
%which will give the highest parallel, from all channels, bit stream.
RESULTS=[];
ChnPwr=[];
RQ=[];
R8=[];
R16=[];
JS=[];
EbNo_AdaptedQQ=[];
EbNo_Adapted88=[];
EbNo_Adapted1616=[];
SumOfNoises=[];
Ssgnl=[];
S11=[];
BEST_RESULT=[];
BEST_RESULT_PWR=[];
BEST_RESULT_PWR_MS=[];
p=0.96; %Gives the percentage of power included  into the main and two side lobes, in the transmitted bandwidth.
for Channels=Qchannels:-1:1 %This loop investigates which is the optimum solution reducing in each cycle the
%number of channels from n to 1, in step of 1.
```

```
RbQPSK=0;
Rb8PSK=0;
Rb16PSK=0;
Stotal=10^-3;% Gives the total received power in the input of the receiver.
if Channels>1
        QN=QNoisepower(1,2:Channels);
else
        QN=0;
end
SumOfNoise=1+(sum(QN)/QNoisepower(1));
SumOfNoises=[SumOfNoises,SumOfNoise]; %CHECK POINT
S1=Stotal/SumOfNoise; %Application of  equation (22). Calculates only S1
                      %in order to calculate the Rb for various modulations.
S11=[S11,S1];%CHECK POINT. Each time the number of channels reduces,
          %the sum(QN) reduces, the variable SumOfNoise reduces, and
          %the allocation of power for each channel (here for the lowest noise channel) increases.
S=p*S1;   %Only the main and the 2 side lobes
          %of the received signal, are of interest. They contain 96%
          %of its total power.
Ssgnl=[Ssgnl,S];    %CHECK POINT
J=QNoisepower(1);   %Interference plus AWGN in the lowest noise channel.
JS_Measured=10*log10(J/S); %Interference plus AWGN to signal power ratio
        %(both of them are referred to the lowest noise channel.)
        %From this ratio the appropriate
        %curve in figures 35-37 is chosen, which for specific Pb
        %gives the required Eb/No.
JS=[JS,JS_Measured];%CHECK POINT. Reducing the number of channels,
        %the allocation of power to each channel increases which in turn
        %decreases the ratio J/S so its log becomes more
        %negative.


% QPSK MODULATION
JS_Theor_QPSK=[-24.5,-15.26,-7.37,-6.9,-5.63,-5.47,-5.32,-5.23,-5,-4.95,-4.75,-4.7,-4.65,-4.59,-4,54];
%From figure 35.
EbNoQPSK=[14,14.5,15,15.5,16,16.5,17,17.5,18,20,21,22,24,26,29];
EbNoQPSK=10.^(0.1.*EbNoQPSK);
%From figure 35, for each separate curve J/S,for Pb=10^-5,
%the ratio Eb/No is derived.
JS_Ratio_AdaptedQ=JS_Theor_QPSK(min(find(JS_Theor_QPSK>=JS_Measured)));
```

```
%Based on the measured J/S, the  curve from figure 35 that fits better, is chosen.
EbNo_AdaptedQ=EbNoQPSK(min(find(JS_Theor_QPSK>=JS_Measured)));
%Finds which theoretical ratio Eb/No from figure 35, fits better to the
%measured J/S.
EbNo_AdaptedQQ=[EbNo_AdaptedQQ,EbNo_AdaptedQ];%CHECK POINT
RbQPSK=(S/No)/EbNo_AdaptedQ; %Application of equation (23).
RQ=[RQ,RbQPSK]; %CHECK POINT. The bit rate for QPSK modulation increases as the number of
        %channels decreases. The cause for this is the increase in S and the decrease
        %in EbNo_AdaptedQ, because, as S increases the log(J/S)
        %becomes more negative and corresponds to curves that
        %lie more left, which in turn give for fixed BER,smaller EbNoQPSK.
        %This matrix (RQ) gives  what the code proposes for bit rate, using the
        %equation 23.Here is not examined if the bandwidth of channel can
        %transmit this bit rate.
if RbQPSK>=bw/2 %It is executed if the bandwidth of channel can
            %not transmit the proposed by equation 23 bit rate.
        RbQPSK=bw/2;
        % 8PSK MODULATION (The variables have the same
        %meaning as in  QPSK modulation but they are referred to Figure 36)
        JS_Theor_8PSK=[-26.7,-20.7,-18.6,-17.3,-16.3,-15.6,-14.8,-14.15,-9.14,-7.35,-7];
        EbNo8PSK=[18,18.5,19,20,21,21.5,22,23,24,26,29];
        EbNo8PSK=10.^(0.1.*EbNo8PSK);
        JS_Ratio_Adapted8=JS_Theor_8PSK(min(find(JS_Theor_8PSK>=JS_Measured)));
        EbNo_Adapted8=EbNo8PSK(min(find(JS_Theor_8PSK>=JS_Measured)));
        EbNo_Adapted88=[EbNo_Adapted88,EbNo_Adapted8];
        Rb8PSK=(S/No)/EbNo_Adapted8;
        R8=[R8,Rb8PSK]; %Because the same ratio J/S for higher modulation scheme is referred to
                %curves that lie more right, these curves for fixed Pb, give bigger
                %Eb/No. From 23  smaller bit rate is taken.
        if Rb8PSK>=0.75*bw
            Rb8PSK=0.75*bw;
            % 16PSK MODULATION(The variables have the
            %same meaning as in modulation QPSK but they are referred to Figure 37.)
            JS_Theor_16PSK=[-26.9,-25.6,-23.55,-22.95,-22.4];
            EbNo16PSK=[24,24.5,27,29,30];
            EbNo16PSK=10.^(0.1.*EbNo16PSK);

            JS_Ratio_Adapted16=JS_Theor_16PSK(min(find(JS_Theor_16PSK>=JS_Measure
            d)));
            EbNo_Adapted16=EbNo16PSK(min(find(JS_Theor_16PSK>=JS_Measured)));
            EbNo_Adapted1616=[EbNo_Adapted1616,EbNo_Adapted16];
```

```
                    Rb16PSK=(S/No)/EbNo_Adapted16;
                    R16=[R16,Rb16PSK];
                    if Rb16PSK>=bw
                        Rb16PSK=bw;
                    end
                end
            end

            RB=[RbQPSK,Rb8PSK,Rb16PSK];
            R=[Channels;RbQPSK;Rb8PSK;Rb16PSK;Channels*max(RB)];
            RESULTS=[RESULTS,R]; %Gives the results coming from different number of channels for
                %different kind of modulation schemes. Basically gives the results coming from execution
                % of procedure described by Figure 38.
            i=(max(RESULTS(5,:)));
            q=find(RESULTS(5,:)==i);
            BEST_RESULT=RESULTS(:,q); %Gives the best result i.e. how many channels, what Rb,
                %what modulation scheme have to be used, in order to achieve
                %the maximum parallel data stream output.



% Allocation of power for each channel.(This belongs in the above FOR LOOP.)
            for U=1:Channels
                CP=S1/QNoisepower(1)*QNoisepower(U);
            ChnPwr=[ChnPwr,CP]; %Allocation of power for each channel,
                %for different number of channels.
            end
end
ii=sum(RESULTS(1,[1:q-1]));
BEST_RESULT_PWR=ChnPwr(ii+1:ii+1+BEST_RESULT(1,:)-1);
BEST_RESULT_PWR=reshape(BEST_RESULT_PWR,length(BEST_RESULT_PWR),1);
%Having found the
%optimum solution, this variable gives how to allocate the total
%received power to each one of the channels (application of equation 19)


%CREATION OF THE SIGNAL THAT IS GOING TO BE
%TRANSMITTED ACCORDING TO THE PRECEDING ESTIMATION.
%Below the variables, put equal to zero, in order to
```

```
%achieve their initialization during each new loop of "while".
RMPSK=0;
RMPSK=max(BEST_RESULT(2:4));%Gives the required bit rate  according to the above estimation.
kM=0;
kM=find(BEST_RESULT(2:4)==RMPSK)+1; %Gives the modulation (k) .
MPSK=0;
MPSK=2^kM; %Gives the type of MPSK. i.e.QPSK,8-PSK or 16-PSK
C=[];
C=frequencyareas(1:BEST_RESULT(1));
CARRIERS=[];
CARRIERS=C+bw/2*ones(size(C));%Gives the carrier frequencies
                                        %in which  transmission has to take place.
AMPLITUTES=[];
AMPLITUTES=sqrt(2*BEST_RESULT_PWR); %Gives the amplitude of signal
                                                %for each separate channel in the input of the receiver.
RSMPSK=0;
RSMPSK=(RMPSK/kM); %Symbol Rate.
fsinitial=0;
fsinitial=fsclock/IntFactor  %Initial Sampling frequency of signal,
                                %prior to interpolation.
samples_per_symbol=0;
samples_per_symbol=round(fsinitial/RSMPSK);%Samples per symbol, prior to interpolation.
number_of_symbols=0;
number_of_symbols=round(L/(IntFactor* samples_per_symbol));%Number of symbols.
InitialPhase=0;
InitialPhase=pi/(2^kM);%Initial phase.

%The below loop gives the files with the symbols that are going to be
%transmitted, different files-symbols for each different channel. Additionally it gives
%the modulated signal for each separate channel(matrix Tx).
ERRORS=0;
TRANSMITTED_SYMBOLS=0;
for tso=1:round(10^6/(number_of_symbols*BEST_RESULT(1)));
waitTitle=sprintf('Calculating BER for Symbol-stream=%2d from %2d',tso,round(10^6/(number_of_symbols*BEST_RESULT(1))));
h=waitbar(0,waitTitle);
XMPSK=[]; %Between each loop "tso",  all the variables, are set equal to zero, in order to transmit new set of %symbols.
YMPSK=[];
Tx=[];
```

```
RxData=[];
aa=[];
for i=1:BEST_RESULT(1)
        waitbar(i/BEST_RESULT(1),h);
        data=[];
        data=randint(1,number_of_symbols,MPSK);
        XMPSK=[XMPSK;data];%The transmitted symbols using n different channels.
        DataMat=[];
        for j=1:samples_per_symbol
        DataMat=[DataMat; data]; %The sampled symbols.
        end;
        IBB=[];
        QBB=[];
        IBB=reshape(cos(2*pi*DataMat/(2^kM)+InitialPhase),1,number_of_symbols*samples_per_symbol);
        %Baseband signal I
        QBB=reshape(sin(2*pi*DataMat/(2^kM)+InitialPhase),1,number_of_symbols*samples_per_symbol);
        %Baseband signal Q

        %Interpolation using FIR square root raised cosine filter till fs=93MHz
        FiltOrdInt=OrderOfFilter(fsinitial*IntFactor,IntFactor,false);%This function determines the order
                                                                      % that will be used by
                                                                      %the square root raised cosine filter
                                                                      %(see footnote 4).
        IBBIntrpFiltr=[];
        QBBIntrpFiltr=[];
        IBBIntrpFiltr=1.1*IntFactor*interpolatesrc(IBB, IntFactor, FiltOrdInt, fsinitial*IntFactor, false);
        %Interpolated signal I
        QBBIntrpFiltr=1.1*IntFactor*interpolatesrc(QBB, IntFactor, FiltOrdInt, fsinitial*IntFactor, false);
        %Interpolated signal Q
        %The interpolatesrc is the function that executes the interpolation.
        %Modulation  of interpolated signal
        n2=0:number_of_symbols*samples_per_symbol*IntFactor-1;
        TxData=[];
        TxData=AMPLITUTES(i).*IBBIntrpFiltr.*cos(2*pi*CARRIERS(i)/fsclock*n2)-
        AMPLITUTES(i).*QBBIntrpFiltr.*sin(2*pi*CARRIERS(i)/fsclock*n2);
        Tx=[Tx;TxData];%Transmitted I,Q,Data using n different channels.
end
LL=length(Tx);

dfr=L-LL;%In order to sum signal and interference,
```

```
        %they have  to have the same number of points.
if dfr<0
        aa=zeros(1,dfr*(-1));
        y12=[y12,aa];
elseif dfr>0
        y12(:,[LL+1:L])=[];
end

%Received signals, plus interference, plus AWGN.  (FFT)
if BEST_RESULT(1)==1
RxData=Tx+y12; % Received Signal plus noise plus interference.
else
RxData=sum(Tx)+y12;     %     "
end

%Demodulation in baseband
n3=0:length(RxData)-1;
for i=1:BEST_RESULT(1)
        I=[];
        Q=[];
        I=RxData.*cos(2*pi*n3*CARRIERS(i)/fsclock+InitialPhase);% Coherent Demodulation
        Q=-RxData.*sin(2*pi*n3*CARRIERS(i)/fsclock+InitialPhase);

        %CIC Decimation
        M=1; %Number of delays.(Default value).
        N=5; %Number of stages.(Default value).
        RCIC=4; %Smallest decimation factor.
        quant=quantizer;
        Idec=[];
        Qdec=[];
        Idec=cicdecimate(M,N,RCIC,I,quant)/((RCIC*M)^N); %Decimated I signal after the CIC filter.
        Qdec=cicdecimate(M,N,RCIC,Q,quant)/((RCIC*M)^N); %     "    Q    "

        %Decimation using Square root raised cosine  filter. This filter represents the filter in FCE.
        %Decimation factor is given in the beggining of the loop while.
        fsDecIn=fsclock/RCIC;%Sampling frequency of signal before second decimation from FCE filter.
        DecSamplesPerSymbolIn=round(samples_per_symbol*IntFactor/RCIC);%Samples per symbol
                                                                %prior second decimation.
        FiltOrdDec=OrderOfFilter(fsDecIn,DecimationFactor,false);%This function determines the order
                                                        %of the filter that will be used by
```

```
                                        %the below square root raised cosine filter
                                        %(see footnote 4).
DecI=[];
DecQ=[];
DecI=decimatesrc(Idec, fsDecIn,  DecimationFactor, number_of_symbols, DecSamplesPerSymbolIn, FiltOrdDec, false);
DecQ=decimatesrc(Qdec, fsDecIn,  DecimationFactor, number_of_symbols, DecSamplesPerSymbolIn, FiltOrdDec, false);
%The function “decimatesrc” executes the decimation,using square root raised cosine filter.

%Integrate
        BaseSamples=floor(samples_per_symbol*IntFactor/(RCIC*DecimationFactor));
        %The remainder number (after the two decimations) of samples per
        %symbol that will be used in the output of the receiver in
        %order to decide about the received symbol.
        Remainder=samples_per_symbol*IntFactor/(RCIC*DecimationFactor)-BaseSamples;
        IntSamples=BaseSamples-1; %Take care of all samples except for the
                %first one (it belongs to transition region that intervenes between two
                %symbols.)
        Offset=1;
        SumI=[];
        SumQ=[];
        SI=[];
        SQ=[];
        SuI=[];
        SuQ=[];
        j=0; % Initial Offset. During the loop it changes
        i=1; % Counter for the position of matrix
        Remainder1=0;
        while i<length(DecI) % This loop is used in case the number
                        % of samples per symbol is not integer.
        j=j+Remainder+Remainder1;
        if j<Offset
        SI=DecI(i:i+BaseSamples-1);
        SQ=DecQ(i:i+BaseSamples-1);
        SuI=sum(SI(:,BaseSamples-IntSamples+1:BaseSamples))/(IntSamples);
        SuQ=sum(SQ(:,BaseSamples-IntSamples+1:BaseSamples))/(IntSamples);
        i=i+BaseSamples;

        else
```

```
SI=DecI(i:i+BaseSamples);
SQ=DecQ(i:i+BaseSamples);
SuI=sum(SI(:,BaseSamples-IntSamples+2:BaseSamples+1))/(IntSamples);
SuQ=sum(SQ(:,BaseSamples-IntSamples+2:BaseSamples+1))/(IntSamples);
i=i+BaseSamples+1;

Remainder1=j-Offset;
j=0;
end
SumI=[SumI,SuI];
SumQ=[SumQ,SuQ];
end

%Calculate phases and correct
RxPhi=[];
RxPhi=atan2(SumQ,SumI); %Based on the magnitudes of I,Q the phase is found.
RxPhi(find(RxPhi<0))=RxPhi(find(RxPhi<0))+2*pi;% Negative phases become positive
%adding 2*pi.
RxPhi=round(RxPhi*(2^(kM-1))/pi);
RxPhi(find(RxPhi==(2^kM)))=0;
YMPSK=[YMPSK,RxPhi];%Gives the received symbols from each separate channel.

end %End of stage of decision, for the n signals, coming from n different spectral areas.


%Calculate Errors
YMPSK=reshape(YMPSK,number_of_symbols,BEST_RESULT(1));
YMPSK=YMPSK';
[number,Ratio]=symerr(XMPSK,YMPSK); %Compares the transmitted and the received symbols,
                                    %defines the number of errors and the
                                    %ratio of total errors over the
                                    %total number of transmitted symbols.
ERRORS=ERRORS+number
TRANSMITTED_SYMBOLS=TRANSMITTED_SYMBOLS+number_of_symbols*BEST_RESULT(1)
y12=[];
y12=y1+y2+y3+ynoise; %Sum of the  Interferences and AWGN
close(h);
end  %End of "tso" loop

BER=(ERRORS/(tso*(number_of_symbols*BEST_RESULT(1))))/kM
```

```
BEST_RESULT_STORY=[BEST_RESULT_STORY,BEST_RESULT];
PSDPlot(RxData, fsclock, 'TRANSMITTED+INTERFERENCE+AWGN signal', false);
BER_FINAL=[BER_FINAL,BER]
end %End of while loop.

spectrum=fft(Tx(1,:));     %Power of transmitted signal in the lower noise channel.
SignalPSD=abs(spectrum).^2/(fsclock*L);
Sgn=sum(SignalPSD(round(L*CARRIERS(1)/fsclock-
2*L*RSMPSK/fsclock):round(L*CARRIERS(1)/fsclock+2*L*RSMPSK/fsclock)));
%Power of signal in the bandwidth of signal referred to the lower noise channel.

dfr=L-LL;% In order to achieve the same number of points
        %for signal and interference and sum them.
if dfr<0
        aa=zeros(1,dfr*(-1));
        y12=[y12,aa];
elseif dfr>0
        y12(:,[LL+1:L])=[];
end



if BEST_RESULT(1)==2
        Interference=Tx(2,:)+y12;
elseif BEST_RESULT(1)==1
        Interference=y12;
else
        Interference=y12+sum(Tx(2:BEST_RESULT(1),:));
end
spectrum=fft(Interference);
InterPSD=abs(spectrum).^2/(fsclock*L);
Jn=sum(InterPSD(round(L*CARRIERS(1)/fsclock-
2*L*RSMPSK/fsclock):round(L*CARRIERS(1)/fsclock+2*L*RSMPSK/fsclock)));
%Power of interference in the bandwidth of signal referred to the lowest noise
%channel. Here the interference does not depend only on the AWGN and the
%initial interferences in the operating bandwidth of the transmitter,
%but also on the side lobes coming from the transmitted signals,
%in the other lower noise channels.

RATIO=10*log10(Jn/Sgn)
```

G. MATLAB_CODE_4 (FUNCTIONS)

The functions Interpolatesrc,Decimatesrc,PSDPlot as in MATLAB_CODE_2.

FUNCTION ORDEROFFILTER

```
function [FiltOrdDec]=OrderOfFilter(fsIn, IntFactor, displayFilter)


%FIR Filter design
rp = 1;          % Passband ripple
rs = 60;         % Stopband ripple
fdec = [fsIn/(4*IntFactor) fsIn/(2*IntFactor)];   % Pass band-cutoff frequencies
a = [1 0];       % Desired amplitudes
% Compute deviations
dev = [(10^(rp/20)-1)/(10^(rp/20)+1) 10^(-rs/20)];
[ndec,fo,ao,w] = remezord(fdec,a,dev,fsIn);
LPF = remez(ndec,fo,ao,w);%Gives the coefficients of the filter.
FiltOrdDec=length(LPF); %Gives the order of the filter that will be used by the
                                %square root raised cosine filter.(see footnote 4).
```

LIST OF REFERENCES

[1] Baylis, Charles P., Software Radio, "The Communications Method of Tomorrow", Department of Electrical Engineering, University of South Florida, Tampa, Florida.

[2] Reed, Jeffrey H., *Software Radio. A Modern Approach to Radio Engineering*, Prentice Hall, Communication Engineering and Emerging Technologies Series, 2002.

[3] Site of SDR Forum (http://www.sdrforum.org).

[4] Intersil, *Technical Manual of Intersil for Four Channel Programmable Digital Down Converter , Model ISL 5216,* Data Sheet February 2002, FN6013.1.

[5] Red River Company, Hardware, *Reference Manual of Red River Card*, Publication Number REF-303-000-R01, 2003.

[6] Intersil, *Technical Manual of Intersil for Four Channel Programmable Digital Up Converter, Model ISL 5217,* Data Sheet March 2002, FN6004.1.

[7] Thesis of Cpt. Georgios Zafeiropoulos, *Software Defined Radio Datalink Implementation using PC-Type Computers*, Naval Postgraduate School, 2003.

[8] Gallager, Robert G., *Information Theory and Reliable Communications*, New York Wiley, 1967.

[9] Robertson, Clark, *Notes in Digital Communications (EC 4550)*, Naval Postgraduate School, Monterey, California.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, VA

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, CA

3. Professor D.C.Boger
   Chair, Information Sciences Dept.
   Naval Postgraduate School
   Monterey, CA

4. Professor Jovan Lebaric
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, CA

5. Embassy of Greece
   Defense and Military Attaché Office
   Washington D.C.

6. Nikolaos Apostolou
   Ano Vathi Pigadaki
   Samos, Greece
   T.K.